

AFML-TR-67-121
PART III

AD0709885

OFFICIAL FILE COPY
EVALUATION OF MOLECULAR WEIGHT
FROM EQUILIBRIUM SEDIMENTATION

PART III. MOLECULAR WEIGHT DISTRIBUTIONS FROM
EQUILIBRIUM SEDIMENTATION-DIFFUSION DATA
VIA LINEAR PROGRAMMING

ROBERT R. JURICK
DONALD R. WIFF
MATATIAHU GEHATIA

TECHNICAL REPORT AFML-TR-67-121, PART III

MAY 1970

This document has been approved for public release and sale;
its distribution is unlimited.

AIR FORCE MATERIALS LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

2004 0302204

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AFML-TR-67-121
PART III

EVALUATION OF MOLECULAR WEIGHT FROM EQUILIBRIUM SEDIMENTATION

PART III. MOLECULAR WEIGHT DISTRIBUTIONS FROM EQUILIBRIUM SEDIMENTATION-DIFFUSION DATA VIA LINEAR PROGRAMMING

ROBERT R. JURICK

DONALD R. WIFF

MATATIAHU GEHATIA

This document has been approved for public release and sale;
its distribution is unlimited.

FOREWORD

This report was prepared by the Polymer Branch of the Nonmetallic Materials Division. The work was initiated under Project No. 7342, "Fundamental Research on Macromolecular Materials and Lubrication Phenomena." Task No. 734203, "Fundamental Principles Determining the Behavior of Macromolecules," with Dr. M. T. Gehatia acting as task scientist. Coauthors are Mr. R. R. Jurick, ASD Computer Science Center (ASVC), and Dr. D. R. Wiff, Research Institute, University of Dayton. The work was administered under the direction of the Air Force Materials Laboratory, Directorate of Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

The report covers research conducted from September 1968 to August 1969. The manuscript was released by the authors in October 1969 for publication as a technical report.

This technical report has been reviewed and is approved.



WILLIAM E. GIBBS
Chief, Polymer Branch
Nonmetallic Materials Division
Air Force Materials Laboratory

ABSTRACT

Within the past decade easy access to high speed digital computers has renewed interest in deriving molecular weight distributions from sedimentation-diffusion equilibrium data. One of the computational schemes which appears most promising is the Simplex Method of linear programming. The purpose of this work was to investigate the advantages and limitations of this approach.

It was found that, even though inferring a molecular weight distribution from sedimentation-diffusion equilibrium data is mathematically an ill-posed problem, the method of linear programming yields qualitatively a good molecular weight distribution. Also, the method proved satisfactory for the case when sedimentation equilibrium data was acquired from only a single angular velocity.

TABLE OF CONTENTS

| SECTION | PAGE |
|---|------|
| I INTRODUCTION | 1 |
| II THEORY | 3 |
| III VARIABLE FACTORS OF COMPUTATION AND THEIR INFLUENCE ON RESULTING MWD | 9 |
| A. Formulation of Computer Problem | 9 |
| B. Distributions Studied | 11 |
| IV DESCRIPTION OF COMPUTER PROGRAM | 14 |
| A. Flow Diagram | 14 |
| B. Description of Variables | 15 |
| C. IN 1 Routine | 16 |
| D. IN 2 Routine | 17 |
| E. LP Solver | 18 |
| F. Plot Routine | 20 |
| V CONCLUSION | 21 |
| APPENDIX: Listing of Computer Program | 25 |
| REFERENCES | 53 |

ILLUSTRATIONS

| FIGURE | | PAGE |
|--------|---|------|
| 1. | Resulting MWD Using Ten Sets of Molecular Weights and $g = 3.0$ (Compare with Figures 2-4) | 55 |
| 2. | Resulting MWD Using Ten Sets of Molecular Weights and $g = 4.0$ (Compare with Figures 1, 3, and 4) | 57 |
| 3. | Resulting MWD Using Twenty Sets of Molecular Weights and $g = 3.5$ (Compare with Figures 1, 2, and 4) | 59 |
| 4. | Resulting MWD Using Twenty Sets of Molecular Weights and $g = 2.5$ (Compare with Figures 1-3) | 61 |
| 5. | The Effect of Decreasing the MWR on the MWD | 63 |
| 6. | The Effect of Decreasing the MWR and the Number of Molecular Weight Sets on the MWD | 64 |
| 7. | The Effect of Further Decreasing the Number of Molecular Weight Sets | 65 |
| 8. | The Effect of Using More Experimental Data on the MWD | 67 |
| 9. | A Typical Curve of Concentration Gradient from One Angular Velocity | 68 |

TABLES

| TABLE | | |
|-------|--|----|
| I. | Exponents of Elements in the Matrix Presented to LP Solver Routine Using Equation 9 | 23 |
| II. | Exponents of Elements in the Matrix Presented to LP Solver Routine Using Equation 20 | 23 |

SECTION I

INTRODUCTION

The relationship describing sedimentation-diffusion equilibrium of an ideal polydisperse solution in an ultracentrifuge can be given by a Fredholm integral equation of the first kind (Reference 1). Since no rigorous solution of this integral is known, there have been many attempts to solve it by approximation (References 2, 3). These efforts mainly involved use of Fourier transforms or Laplace transforms by assuming an approximate functional expression for the experimental concentration gradient along the ultracentrifugal cell, or by expanding the molecular weight distribution (MWD) into a polynomial of assumed functions.

The main weakness has been that some parts of the calculated distribution would be negative. Physically, of course, we know that the MWD for any molecular weight must always be positive or zero. Recently Lee (Reference 4) carried out an investigation of the Fredholm integral equation and found that mathematically it is an "ill-posed" problem. In trying to infer a MWD from experimental measurements of concentration gradients small errors can lead to an unacceptable MWD. Therefore, we compromised in trying to determine only an "overall" shape of the MWD without being specific to individual points, i. e. , we allowed certain fluctuations of the curve to be present and ignored fine structure.

To generalize a theoretical analysis, let us accept that the MWD can be slightly negative for some molecular weight values. Since we chose to ignore the point-by-point functional form of the MWD, the next logical step would be to subdivide the MWD into narrow (not infinitesimal) but finite molecular weight strips. This would result in approximations of MWD by rectangles of finite width and would lead naturally to the use of matrices. This has been done (Reference 5) but unfortunately the matrices are "ill-conditioned" or nearly singular.

Scholte (Reference 6 and 7) in 1968, still using matrices, applied the scheme of linear programming to infer a MWD from experimental measurements of concentration gradients at various angular velocities. The main

advantage to this approach is that values of the MWD are forced to be greater than or equal to zero and "slack variables" are introduced to account for experimental error. Scholte evaluated the MWD at ten molecular weights, then shifted to ten other molecular weights in a prescribed manner, continuing until, finally, there were four such sets. Since each set represented an individual solution, one quarter of the sum of the four sets also represented a solution. By doing this, Scholte obtained good agreement between his assumed and calculated molecular weight distributions.

There are, however, three reasons why Scholte's scheme cannot be blindly applied to other systems. These are: (1) Scholte dealt with a molecular weight range of 5×10^4 to 10^6 ; by comparison, in many cases of synthetic polymers the range is much narrower, e.g., 0 to 10^5 . (2) Scholte used five or more angular velocities, each requiring several days for equilibrium. There are, however, cases when equilibrium at each velocity requires a much longer time (Reference 8). Therefore, it is important to have a scheme which would produce a MWD from data taken at one velocity. (3) Since our interest was in a different molecular weight range and we were using experimental data from only one angular velocity, the effects of experimental error on the calculated MWD had to be investigated.

A computer program using Scholte's ideas was independently coded and a different linear programming (LP) solving routine was employed. The new program reliability was verified by reproducing Scholte's published results. Then application of the new program to new specific needs stated above were investigated. A brief description of linear programming theory follows.

SECTION II

THEORY

In the brief discussion which follows, all theorems and definitions are given without proof or examples. All material on linear programming was taken from other works (References 9, 10, and 11).

Definition 1. A simplex is an n -dimensional convex polyhedron having exactly $n+1$ vertices. The boundary of the simplex contains simplicial faces of dimension i where $i < n$. The number of such faces of dimension i is $\binom{n+1}{i+1}$ where $\binom{n}{m} = n! / m!(n-m)!$. A simplex in zero dimension is a point, in 1-dimension a line, in 2-dimension a triangle, in 3-dimension a tetrahedron, etc. The equation of a simplex with unit intercept is $X_i \geq 0$ and $\sum_i X_i = 1$.

Definition 2. A subset C of E_n (n -dimensional Euclidean Space) is a convex set if and only if for all pairs of points \underline{V}_1 and \underline{V}_2 in C any convex combination

$$\underline{V} = \beta_1 \underline{V}_1 + \beta_2 \underline{V}_2 \quad (1)$$

is also in C , where β_i are scalars, $\beta_i \geq 0$, and $\sum_i \beta_i = 1$.

Definition 3. A point \underline{V} in a convex set C is called an extreme point if \underline{V} cannot be expressed as a convex combination of any other two distinct points in C . That is, if we denote the convex set of solutions to the linear programming problem by K and if K is a convex polygon, then K is the convex hull of the extreme points of K . Therefore, every feasible solution in K can be represented as a convex combination of the extreme feasible solutions in K .

Theorem 1. The set of all feasible solutions to the linear programming problem is a convex set.

In general, the linear programming problem can be described as follows: Given is a convex set defined by a set of linear constraints in E_n . From all the points belonging to the convex set, we wish to determine a subset of points (which will contain either one or many points) for which a linear objective function is optimized.

Usually we are confronted with a set of simultaneous equations

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1m}x_m = b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nm}x_m = b_n \end{array} \quad (2)$$

where $n > m$. For simplicity let m equal n ; let A be the matrix, $\{a_{ij}\}$, ($i = 1, 2, 3, \dots, m$ and $j = 1, 2, 3, \dots, m$); \underline{X} the vector $[X_j]$, $j = 1, 2, 3, \dots, m$ and \underline{b} the vector $\{b_i\}$, $i = 1, 2, 3, \dots, m$. Then Equation 2 can be written in the form

$$\underline{A}\underline{x} = \underline{b} \quad (3)$$

Since A is a square matrix and assumed nonsingular, the solution vector is expressed as

$$\underline{x} = A^{-1} \underline{b} \quad (4)$$

A simple computational scheme is the complete elimination method of Jordan and Gauss which has a finite number of steps or iterations. In just m iterations the procedure multiplies the system (Equation 2) by A^{-1} to obtain Equation 4. This is the standard matrix problem which is assumed familiar to the reader.

Now, in linear programming the problem is reversed. Instead of having an "over-determined" system as indicated by Equation 2, we have an "under-determined" system (i. e., $n < m$) subject to other constraints. That is, we wish to find a vector $\{x_i\}$, $i = 1, 2, 3, \dots, m$ which minimizes the linear form (i. e., the objective function)

$$C_1x_1 + C_2x_2 + C_3x_3 + \dots + C_mx_m \quad (5)$$

subject to linear constraints

$$x_j \geq 0, \quad j = 1, 2, 3, \dots, m \quad (6)$$

and the set of equations given by Equation 2 but with $n < m$.

For large n and m it would be an impossible task to evaluate all possible solutions and select one that minimizes the objective function. A computational

scheme is desired which converges to a minimum solution. The Simplex Method, devised by Dantzig (Reference 11) is such a scheme. In Reference 11 the equation $\sum X_i$ is used as a constraint. The procedure finds an extreme point and determines whether it is the minimum. If it is not, the procedure finds a neighboring extreme point whose corresponding value of the objective function is less than or equal to the preceding value. In a finite number of such steps (usually between n and $2n$), a minimizing feasible solution is found. The Simplex Method makes it possible to discover whether the problem has any finite minimizing solutions or no feasible solutions at all.

Consideration is now given to how this can be related to the problem at hand, namely, molecular weight determination via sedimentation-diffusion equilibrium. The equation describing sedimentation-diffusion equilibrium for a heterogeneous system is given (from Reference 1) by

$$-\frac{1}{C^0} \frac{dC}{d\xi} = \int_0^\infty \frac{\lambda^2 M^2 e^{-\lambda M} F(M)}{1 - e^{-\lambda M}} dM \quad (7)$$

where,

$$\int_0^\infty F(M) dM = 1 \quad (8)$$

In the above equations C^0 is the concentration of the original solution, C is the equilibrium concentration at radial distance r , M is the molecular weight, $F(M)$ is the frequency function of molecular weight, $\xi = (r_b^2 - r^2)/(r_b^2 - r_m^2)$ with r_m the radial distance from the center of rotation to the meniscus, and r_b the radial distance from the center of rotation to the bottom of the cell. Also, $\lambda = (1 - v\rho) \omega^2 (r_b^2 - r_m^2)/2RT$, where v is the partial specific volume of the dissolved substance, ρ is the density of the solution, ω is the angular velocity in radian per second, R is the gas constant, and T the absolute temperature.

Rewriting Equations 7 and 8 for the discrete case (Dirac δ -functions) one obtains

$$U(\lambda_i, \xi_n) = \sum_m \frac{\lambda_i^2 M_m^2 e^{-\lambda_i M_m \xi_n}}{1 - e^{-\lambda_i M_m}} f_m \quad (9)$$

and,

$$\sum_m f_m = 1 \quad (10)$$

where

$$U(\lambda_i, \xi_n) = - \frac{1}{C^0} \left(\frac{dC}{d\xi_n} \right)_{\lambda_i} \text{ and } f_m$$

is the weight fraction of molecules of a given molecular weight M_m in the original sample. Recall that the $U(\lambda_i, \xi_n)$ and ξ_n are the experimentally measured quantities with λ_i being the product of a constant (determined from auxiliary measurements) and the square of the angular speed of the rotor.

For convenience of notation let

$$K_{\ell n} = \frac{\lambda_i^2 M_m^2 e^{-\lambda_i M_m \xi_n}}{1 - e^{-\lambda_i M_m}} \quad (11)$$

and

$$U_\ell = U(\lambda_i, \xi_n) \quad (12)$$

where for each i , $n = 1, 2, \dots, N$; $i = 1, 2, \dots, I$; $m = 1, 2, \dots, M$; and $\ell = 1, 2, \dots, L$ with $L = IN$ and $L > M$.

Thus Equation 9 becomes

$$U_\ell = \sum_m K_{\ell m} f_m \quad (13)$$

Since the quantities of U_ℓ are experimentally measured, they will in all probability be greater than or less than their true precise value (i.e., there exists experimental error). Although this physical fact is accepted, experimentally Equation 13 does not hold true. This is especially apparent when we investigate the matrix $\{K_{\ell m}\}$ and find it ill-conditioned. In essence, the matrix $\{K_{\ell m}\}^{-1}$ acts as an "amplifier" for any error which might exist in the set $\{U_\ell\}$.

If we grant that an error in U_ℓ exists, Equation 13 becomes

$$U_\ell = \sum_m K_{\ell m} f_m + \epsilon_\ell \quad (14)$$

where ϵ_ℓ is the experimental error in U_ℓ . Since the application of linear programming necessitates that all x_i (see Equation 1) are positive or zero, we must account for error's being positive or negative. It is the inclusion of error that now enables us to go from an "over determined" system to an "under determined" system. The linear programming procedure is now applicable. In particular, you will recall that a modified Simplex Method can be used.

Recapitulating, we now obtain the formulation of the linear programming scheme as used to determine the MWD from sedimentation-diffusion equilibrium. We wish to find the set $\{f_m\}$, $m = 1, 2, \dots, Q$, which minimizes the linear form (i. e., the objective function)

$$\sum_{\ell=1}^L (\delta_\ell + \beta_\ell) \quad (15)$$

subject to the linear constraints

$$\left. \begin{aligned} f_m &\geq 0, \quad m = 1, 2, 3, \dots, Q \\ \delta_\ell &\geq 0 \\ \beta_\ell &\geq 0 \end{aligned} \right\}, \ell = 1, 2, 3, \dots, L \quad (16)$$

and

$$\begin{aligned} K_{11}f_1 + K_{12}f_2 + \dots + K_{1Q}f_Q + \delta_1 - \beta_1 + 0 + 0 + \dots + 0 + 0 &= U_1 \\ K_{21}f_1 + K_{22}f_2 + \dots + K_{2Q}f_Q + 0 + 0 + \delta_2 - \beta_2 + \dots + 0 + 0 &= U_2 \\ \vdots & \\ K_{L1}f_1 + K_{L2}f_2 + \dots + K_{LQ}f_Q + 0 + 0 + 0 + 0 + \dots + \delta_L - \beta_L &= U_L \end{aligned} \quad (17)$$

where $L > Q^{**}$. Let the set $\{x_i\}$, $i = 1, 2, 3, \dots, Q + 2L$ be composed of f_m values for $i = 1, 2, 3, \dots, Q$, and alternately δ_ℓ and β_ℓ for $i = Q + 1, Q + 2, \dots, Q + 2L$; where all $x_i \geq 0$. Also let the $L \times (Q + 2L)$ matrix P be represented by

****This is not an absolutely necessary condition. When one velocity was used the situation arose where $L < Q$.**

$$P = \begin{pmatrix} K_{11} & K_{12} & \cdots & K_{1Q} & 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ K_{21} & K_{22} & \cdots & K_{2Q} & 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ K_{L1} & K_{L2} & \cdots & K_{LQ} & 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix} \quad (18)$$

Then, in matrix notation the problem is formulated by

$$P\underline{X} = \underline{U} \quad (19)$$

The next section will describe the application of this method.

SECTION III

VARIABLE FACTORS IN COMPUTATION AND THEIR INFLUENCE ON RESULTING MWD

A. FORMULATION OF COMPUTER PROBLEM

The objective of this section is to present the results which three variable factors investigated have on a MWD. Since the actual programming involved a slight modification of Equation 9, a listing and discussion of the variable factors investigated will be preceded by a discussion of the actual equation programmed.

The programmed equation is given by

$$U(\lambda_k, \xi_i) = \sum_n \frac{\lambda_k^2 M_n^2 e^{-\lambda_k M_n \xi_i} F(M_n) \Delta M_n}{1 - e^{-\lambda_k M_n}} \quad (20)$$

Here all quantities have the same meaning as in Equation 7 and 9. However, we must remember that, since λ is proportional to the square of the angular velocity, the index k indicates the various velocities at which equilibrium was achieved. For each velocity there exists a set of ξ -values, i. e., for each λ_k there is a corresponding set $\{\xi_i\}$. If there are data from five velocities and for each velocity there corresponds five ξ -values, this would imply twenty-five U -values. It is imperative that the molecular weight range being investigated incorporate all molecular weights present in the solution sample being centrifugated. Since Equation 9 and 20 deal with discrete molecular weights, some procedure must be employed to span the entire molecular weight range (MWR). Following the idea of Scholte (References 6 and 7), a multiplicative factor (g -factor) has been introduced. Therefore, starting with the first molecular weight M_1 , the other molecular weights in a given sampled set could be generated. Knowing, a priori, the MWR we can now calculate the g -factor and the number (N_Q) of molecular weights needed to span the MWR. (The g -factor is later used as a variable parameter related to the error in the experimental concentration gradients. However, the value of g must be at least large enough to ensure the actual MWR present in a given experiment

is spanned. A convenient technique for finding the MWR is given in Reference 8). The molecular weights sampled will be

$$M_n = M_1 g^{n-1} \quad (21)$$

where $n = 1, 2, 3, \dots, N_Q$. This enables one to divide the MWR into non-overlapping subranges. Each subrange span is denoted by

$$\Delta M_n = M_n - M_{n-1} \quad (22)$$

where M_0 is assumed zero. By employing the averaging technique of Scholte (Reference 7), after solving Equation 20 for one set of molecular weights, a new set of molecular weights is selected in a prescribed manner. The new set is shifted relative to the previous set by a multiplicative factor $g^{1/N}$, where N is the number of desired molecular weight sets. That is, if the number of the set is labeled by the index n and the molecular weights within a set by j , then

$$M_{j,n} = M_{1n} g^{j-1} \quad (23)$$

and

$$\Delta M_{jn} = (g^{1/2} - g^{-1/2}) M_{jn} \quad (24)$$

where $j = 1, 2, 3, \dots, N_Q$; $n = 1, 2, 3, \dots, N$; and $M_{0,n} = 0$.

The concentration of molecular weights in a given subrange is simply the weight fraction (f_m) multiplied by the initial solution concentration ($C_m^0 = C^0 f_m$). When Scholte (Reference 7) presents his final results they are in the form $MF(M)$ versus M . In this work a modified system $F(M)$ versus M has been calculated (Equation 20), since one usually has less qualitative feeling for $MF(M)$ than for $F(M)$.

Now, it is possible to list the variable parameters investigated in this work. They are as follows:

1. The effect that varying the g -factor (i. e. , the span of the molecular weight subrange) has on the resulting MWD; and also the effect when the number of sets of molecular weights sampled was varied.

2. How the resulting MWD is affected by varying the number of elements in the sets $\{\xi_i\}$ and $\{\lambda_k\}$.

3. Whether the introduction of error into idealized U-values affects the resulting MWD. This includes normal random error and weighted random error.

The results of each will be discussed successively in the following section.

B. DISTRIBUTIONS STUDIED

The g-factor is related to the experimental error in the U-values (Reference 7). Figures 1 through 4 show the effect of varying the g-factor and the number of sets of molecular weights. In each case the solid-line curve is the assumed distribution from which U-values were calculated. For all curves, seven velocities were used and with each velocity five ξ -values. The squared angular velocities were 4.1693×10^5 , 5.8370×10^5 , 8.1718×10^5 , 11.4406×10^5 , 16.0168×10^5 , 22.4235×10^5 , and 31.3929×10^5 rad²/sec² with $\xi = 0$, 1/4, 1/2, 3/4, and 1 for each case. Therefore, thirty-five U-values were used for these calculations.

As previously mentioned, we must always be sure that the MWR is wide enough to incorporate all molecular weights present in the sample. To study the effect of range size on the MWD, all parameters in Figure 4 were held constant except the value for MWR, which was brought closer to the MWR of the assumed distribution. As shown in Figure 5, structure begins to appear when the highest molecular weight sampled was not far beyond the actual highest molecular weight present. Figure 5 represents a calculation involving twenty molecular weight sets. When the number of molecular weight sets was decreased from twenty to ten (Figure 6) then five (Figure 7), there was no appreciable change except that, naturally, the calculated points were spaced farther apart.

What would be the effect if the number of ξ -values associated with each velocity was increased? As previously mentioned, five ξ -values have been used per velocity for Figures 1 through 7. Figure 8 shows the results with all

parameters of Figure 6 held constant except that, now, nine ξ -values were used. The nine ξ -values were so chosen that $\xi = 0$ to 1 with $\Delta \xi = 1/8$. From the study of this assumed distribution it appears that using five ξ -values, sampling twenty sets of molecular weights, and using a g-factor ≈ 2.0 seemed to have produced the optimum desired results, i. e., the calculated MWD agreeing best with the assumed MWD. If the g-factor became too small the result was noise, that is, the accuracy of the U-values did not warrant such precision, or the matrix in the LP solver routine became singular.

At this stage a normal Gaussian distribution with a MWR of 0 to 120,000 was investigated. Once again the g-factor was varied. The lowest g-value used was 1.15 and the highest 4.0. The former resulted in an erratic MWD and the latter resulted in a curve which went exponentially to zero at high molecular weights. The best g-value for this specific case was $g = 1.8$. In general, a satisfactory technique was to start with a low value of g. Then as the value of g was increased the erratic behavior of the MWD disappeared. At the g-value where the erratic results seemed to disappear, that value was established as the appropriate one. Then the maximum reliable "fine structure" for a given set of experimental U-values was attained.

Using this assumed normal distribution (its functional form) the MWR was shifted to investigate the reliability of the method for various molecular weight ranges. One range tried was from 0 to 12,000 and another from 10^5 to 10^6 . In each case the results were satisfactory, considering that in all cases g was kept constant ($g = 1.8$).

As previously mentioned, the U-values used resulted from seven assumed velocities ranging from about 6,000 to 50,000 RPM. It would be advantageous if the experimental U's were obtained from an equilibrium sedimentation-diffusion experiment at only one angular velocity. To check this, the normal Gaussian distribution, MWR from 0 to 120,000, was approximated (holding all other variables constant) by deleting all U-values associated with various angular velocities. All combinations of the velocities were tried. By using only the lowest angular velocity (6,166 RPM), the computer program produced a MWD which "fit" the assumed MWD as well as the case where all seven angular velocities were used. In fact, all single velocity cases resulted in

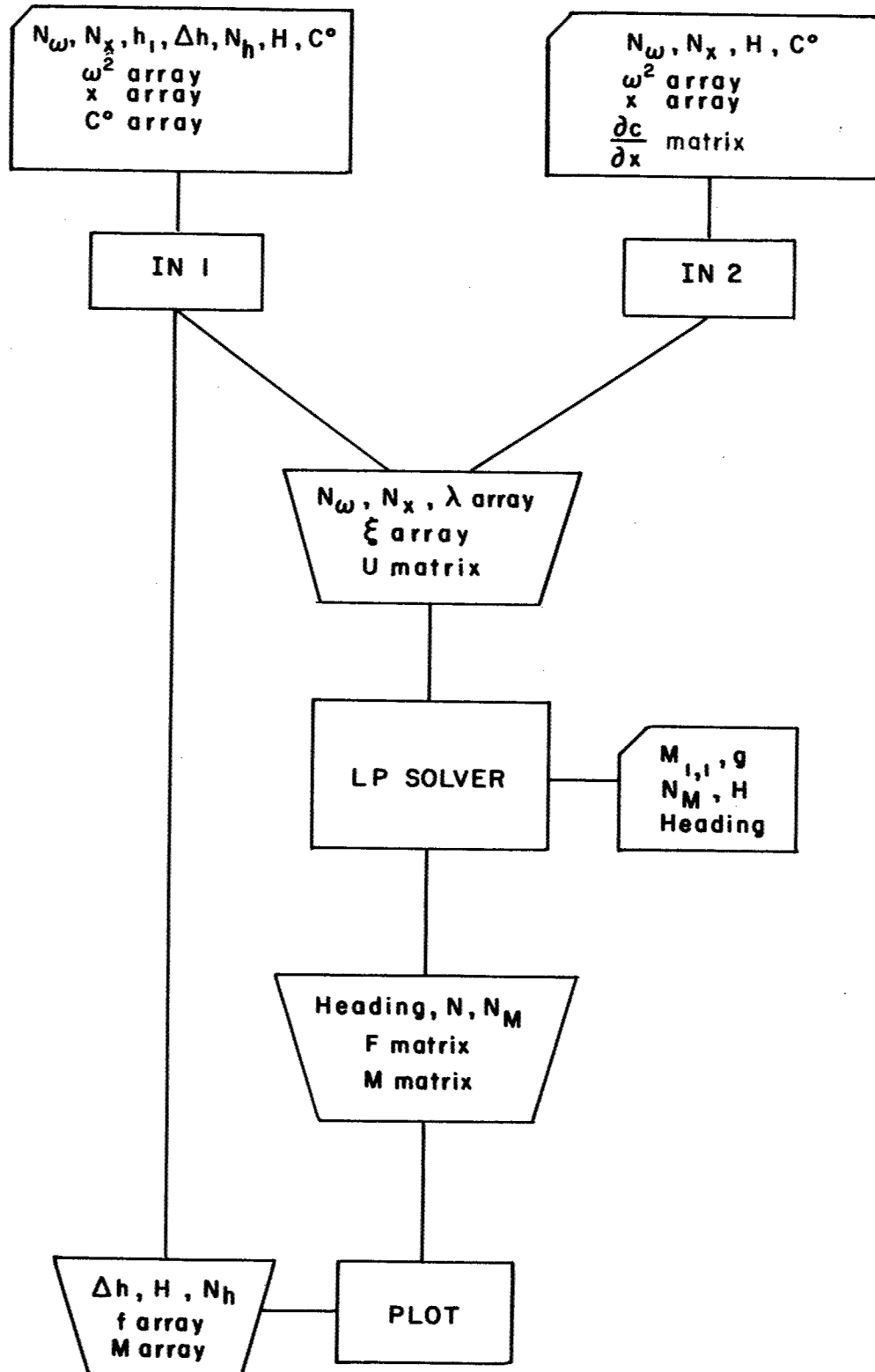
a reasonable MWD. Therefore, we would conclude that an acceptable MWD could be obtained from an equilibrium sedimentation-diffusion experiment at one angular velocity, at least with a MWR of 0 to 120,000.

The third area of investigation involved use of the normal Gaussian MWD ($0 < M < 120,000$) U-values from one angular velocity (6,166 RPM) and at $\xi = 0, 1/4, 1/2, 3/4$, and 1 to find the effect that error in the U-values would have on the calculated MWD. Error (1, 2, 5, 10, and 20%, respectively) was introduced by aid of a random error generator. For each magnitude of error, five calculations were employed to vitiate any wrong conclusions that one error distribution might have on the final MWD. For each case (1 to 20%) when the error was normally distributed (dotted line Figure 9) the calculated MWD agreed with the assumed MWD. Naturally the 1% error case gave the best "fit" to the assumed MWD, but even for the 20% error case the calculated MWD was not unacceptable. When the error introduced in the U-values was such as to be weighted (dashed curves Figure 9), the calculated MWD was entirely different from the expected normal MWD. This phenomenon substantiates the findings of Lee (Reference 5) and Tikhonov and Glasko (Reference 12).

SECTION IV

DESCRIPTION OF COMPUTER PROGRAM

FLOW DIAGRAM



DESCRIPTION OF VARIABLES

| | |
|------------------|---|
| $a_{i,j}$ | - entry in matrix for LP problem |
| b | - largest x value |
| b_i | - right-hand side of LP problem |
| C° | - initial concentration of solution |
| C_m° | - initial concentration of solution whose molecular weight is $\frac{h_n}{H}$ |
| f_n | - weight fraction of molecular weight M_n |
| $F_{j,n}$ | - results of LP solution, frequency list for mol. wt. of $M_{j,n}$ |
| g | - M multiplier |
| h_n | - input test array which is a function of molecular weight (References 5 and 8) |
| H | $= \frac{2RT}{1-v\rho}$ |
| m | - smallest x value |
| M_n | - molecular weight array associated with input test array |
| $M_{j,n}$ | - molecular weight matrix of values used in LP solution |
| N | - number of LP sets to try |
| N_h | - number of input h values |
| N | - number of M values to use for use LP solution |
| N_ω | - number of input ω^2 values |
| N_x | - number of input x values |
| $U_{k,\ell}$ | $= -\frac{1}{C^\circ} \frac{dc}{d\xi}$ |
| x_ℓ | - array of distances squared from center of rotation |
| λ_k | - function of ω_k^2 |
| Δh | - constant difference between values of h_n array |
| $\Delta M_{j,n}$ | - difference between successive j values of $M_{j,n}$ matrix |
| ξ_{ℓ^2} | - function of x_ℓ |
| ω^2 | - square of the angular velocity |

PART III

IN 1 ROUTINE

Input: N_ω N_x , h_1 , Δh , N_n , H , C° , ω_k^2 array, where $k = 1, \dots, N_\omega$
 x_j array, where $\ell = 1, \dots, N_x$
 C_n° array, where $n = 1, \dots, N_h$

$$1. \quad h_{n+1} = h_n + \Delta h \text{ for } n = 1, \dots, (N_h - 1)$$

$$2. \quad m = x_1$$

$$b = x_{N_x}$$

$$\lambda_k = (b - m) \cdot H \cdot \omega_k^2 \text{ for } k = 1, \dots, N_\omega$$

$$3. \quad \xi_\ell = \frac{b - x_\ell}{b - m} \text{ for } \ell = 1, \dots, N_x$$

$$4. \quad M_n = \frac{h_n}{H} \text{ for } n = 1, \dots, N_h$$

$$f_n = \frac{C_n^\circ}{C^\circ}$$

$$5. \quad U_{k,\ell} = \sum_{n=1}^{N_h} \frac{(\lambda_k M_n)^2 e^{-\lambda_k M_n \xi_\ell}}{1 - e^{-\lambda_k M_n}} f_n$$

for $k = 1, \dots, N_\omega$ and $\ell = 1, \dots, N_x$

6. Write out ω^2 array, ξ array, and U matrix

7. Call LP SOLVER

AFML-TR-67-121
PART III

IN 2 ROUTINE

Input: N_ω , N_x , H , C° , ω_k^2 array where $k = 1, \dots, N_\omega$

x_ℓ array where $\ell = 1, \dots, N_x$

$(\frac{dC}{dx})_{k\ell}$ matrix where $k = 1, \dots, N_\omega$ and $\ell = 1, \dots, N_x$

1. $m = x_1$

$b = x_{n_x}$

2. $\lambda_k = (b-m) H \omega_k^2$ for $k = 1, \dots, N_\omega$

3. $\xi_\ell = \frac{b-x_\ell}{b-m}$ for $\ell = 1, \dots, N_x$

4. $U_{k,\ell} = \frac{b-m}{C^\circ} (\frac{dC}{dx})_{k,\ell}$ for $k = 1, \dots, N_\omega$
and $\ell = 1, \dots, N_x$

5. Write out ω^2 array, ξ array, and U matrix

6. Call LP SOLVER

LP SOLVER

1. Input from IN 1: N_ω , N_x , λ array, ξ array, and U matrix
2. Input from cards: $M_{1,1}$, g , N_M , N , HEADING
3. Write out input from cards

$$4. \quad N_{row} = N_\omega : N_x$$

$$N_{col} = N_M + 2 N_{row}$$

$$n = 1$$

5. Calculate the following matrix entries

$$a_{i,j} = 1 \text{ for } i = 1, \dots, N_{row} \text{ and } j = N_M + 1, N_{col}-1, 2$$

$$a_{i,j} = 1 \text{ for } i = 1, \dots, N_{row} \text{ and } j = N_M + 1, N_{col}, 2$$

$$a_{i,j} = \frac{(\lambda_{k^{M_{j,n}}})^2 e^{-\lambda_{k^{M_{j,n}}} \xi_{\ell}}}{1 - e^{-\lambda_{k^{M_{j,n}}}}} \Delta M_{j,n}$$

$$\text{for } i = 1, \dots, N_{row} \text{ and } j = 1, \dots, N_M$$

$$\text{where } k = \left\lfloor \frac{i-1}{N_x} \right\rfloor + 1$$

$$\ell = i(k-1) \cdot N_x$$

$$M_{j,n} = M_{1,n} \cdot g^{j-1}$$

and

$$\Delta M_{j,n} = (g^{1/2} - g^{-1/2}) M_{j,n}$$

6. Calculate the following right-hand sides $b_i = U_{k,\ell}$ where i, k , and ℓ are defined as in step 5.
7. Calculate the following objective coefficients

$$C_j = 0 \text{ for } j = 1, \dots, N_M$$

$$C_j = 1 \text{ for } j = N_M + 1, \dots, N_{col}$$

LP SOLVER

8. Write out the determinants of:

$$\begin{bmatrix} a_{ij} \end{bmatrix} \text{ for } i = 1, \dots, N_M \text{ and } j = 1, \dots, N_M$$

$$\begin{bmatrix} a_{ij} \end{bmatrix} \text{ for } i = N_M + 1, \dots, 2 \cdot N_M$$

$$\begin{bmatrix} a_{ij} \end{bmatrix} \text{ for all sets of } N_M \text{ rows less than } N_{\text{row}}$$

9. Write out matrix $\begin{bmatrix} a_{ij} \end{bmatrix}$ for $i = 1, \dots, N_{\text{row}}$ and $j = 1, \dots, N_M$ in exponent form.

10. Call LP solver and store solutions in $F_{j,n}$ array.

11. Write out input RHS
 computer RHS using solution
 difference of RHS'S
 absolute value of relative differences of RHS
 average absolute relative difference

12. $n \leftarrow n + 1$

$$M_{1,n} \leftarrow (M_{1,n-1})g^{1/N}$$

13. Return to step 5 until n exceeds N

14. Write out $F_{j,n}$ matrix

15. Call PLOT routine

PLOT ROUTINE

1. Input from LP SOLVER: HEADING, N , N_m , F , and M matrices
2. Input from IN 1 (if used): Δh , H , N_n , f , and M arrays
3. $\Delta M = \frac{\Delta h}{H}$
4. Write out heading
5. Label vertical axis $F(M)$
6. Label horizontal axis M
7. Plot $\frac{f_n}{\Delta M}$ versus M_n for $n = 1, \dots, N_h$
8. Plot $F_{j,n}$ versus $M_{j,n}$ for $n = 1, \dots, N$ and $j = 1, \dots, N_M$
9. STOP

SECTION V

CONCLUSION

To date, the linear programming method seems to be one of the most promising schemes for obtaining the molecular weight distribution from sedimentation-diffusion equilibrium data. There are five main features of this investigation which are worthy of mention.

1. The linear programming method has been found to give acceptable results for the case of experimental data obtained at one angular velocity.

2. It was found that if a normal random error of the experimental gradient curve was about 20% the linear programming method produced a MWD with satisfactory precision. However, if the experimental error was weighted, i.e., the concentration gradient curve was distorted from the true curve, a 1 or 2% error led to an absurd molecular weight distribution. This agrees with the findings of Lee (Reference 5), Tikhonov and Glasko (Reference 12), and Tikhonov (References 13 and 14).

3. This investigation did not involve any modification of the LP solver routine. The LP solver limitations were manifested by spurious points sometimes appearing in the determined molecular weight distribution. In general, the linear programming method presents only an overall molecular weight structure. Therefore, when twenty sets of molecular weights were sampled, it was obvious when one point was completely illogical.

4. A great improvement was achieved by solving for $F(M)$ directly (Equation 20), rather than via f_m (Equation 9). In the former case the matrix presented to the LP solver routine was not so ill-conditioned (Tables I and II).

5. The following test was made after each call of the LP solver routine. By having the calculated MWD, the computer program could calculate new U -values (U_{calc}). The difference between U_{exp} and U_{calc} was then printed. Also, the absolute relative differences, the averaged absolute relative error for one set of molecular weights, and the averaged absolute relative error averaged over all molecular weight sets were printed. In general, a good "fit" between the assumed MWD and the derived MWD showed low values for

all the above error analyses; however, the converse was not always found to hold true. At present much effort is being focused on determination of a procedure for obtaining a one-to-one correspondence between the error analysis criteria and the "fit" of the derived MWD.

TABLE I
EXPONENTS OF ELEMENTS IN THE MATRIX PRESENTED TO
LP SOLVER ROUTINE USING EQUATION 9*

| | | | | | | | | | |
|----|----|----|----|---|---|---|---|----|----|
| -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | -5 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -3 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 |

* Value of determinant = -0.3932972 E-31

TABLE II
EXPONENTS OF ELEMENTS IN THE MATRIX PRESENTED TO
LP SOLVER ROUTINE USING EQUATION 20*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 |
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 |
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 6 |
| 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 6 |
| 1 | 1 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 |
| 1 | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 0 |
| 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 4 | 2 |
| 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 | 4 |
| 1 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 6 | 6 |
| 1 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 |

* Value of determinant = -0.1500715 E 17

APPENDIX
LISTING OF COMPUTER PROGRAM

AFML-TR-67-121
PART III

| | |
|--|----------|
| \$IBETC EXEC. DECK | EXEC.001 |
| C | EXEC.002 |
| C MAIN PROGRAM EXECUTIVE CONTROL | EXEC.003 |
| C INITIALIZE PLOTTING ROUTINES | EXEC.004 |
| C INITIALIZE PLOT COUNT | EXEC.005 |
| C DETERMINE SEQUENCE OF SUBROUTINES CALLED | EXEC.006 |
| C TERMINATE PLOTTING BEFORE EXITING | EXEC.007 |
| C WRITE HEADING AND PLOT COUNT | EXEC.008 |
| C | EXEC.009 |
| COMMON /PLTR/ PDATA(438), IPLTS, HEAD(12) | EXEC.010 |
| COMMON /PRTCTL/ WRITE | EXEC.011 |
| C | EXEC.012 |
| LOGICAL WRITE | EXEC.013 |
| C | EXEC.014 |
| IPLTS = 0 | EXEC.015 |
| CALL PLOTS (PDATA, 438) | EXEC.016 |
| 1 READ (5,500) HEAD | EXEC.017 |
| WRITE (6,600) HEAD | EXEC.018 |
| CALL INI (\$900) | EXEC.019 |
| IF (WRITE) WRITE (6,600) HEAD | EXEC.020 |
| CALL LPS (\$900) | EXEC.021 |
| CALL PLOTR | EXEC.022 |
| GO TO 1 | EXEC.023 |
| 900 WRITE (6,601) IPLTS | EXEC.024 |
| CALL PLOTE | EXEC.025 |
| STOP | EXEC.026 |
| 500 FORMAT (12A6) | EXEC.027 |
| 600 FORMAT (1H1, 12A6 / (1X, 12A6)) | EXEC.028 |
| 601 FORMAT (1H0, 12, 17H PLOT(S) COMPLETE) | EXEC.029 |
| END | EXEC.030 |

AFML-TR-67-121
PART III

| | |
|--|----------|
| \$IBFEC BIN1. DECK | BIN1.001 |
| C | BIN1.002 |
| C BLOCK DATA SUBPROGRAM TO SUPPLY INPUT DATA WHEN INP1. IS BY-PASSED | BIN1.003 |
| C | BIN1.004 |
| C BLOCK DATA | BIN1.005 |
| C | BIN1.006 |
| C COMMON /BIN1/ NW, NX, XL(20), Z(20), U(20,20) | BIN1.007 |
| C | BIN1.008 |
| DATA NW /5/ | BIN1.009 |
| DATA NX /5/ | BIN1.010 |
| DATA XL / 2.5E-6, 10.E-6, 40.E-6, 160.E-6, 640.E-6, 15*0. / | BIN1.011 |
| DATA Z /1.0, .75, .5, .25, 0., 15*0. / | BIN1.012 |
| DATA U / .187, .405, .372, .144, .005, 15*0., | BIN1.013 |
| X .208, .554, .639, .337, .026, 15*0., | BIN1.014 |
| X .232, .799, 1.346, .959, .164, 15*0., | BIN1.015 |
| X .260, 1.237, 0., 0., 0., 15*0., | BIN1.016 |
| X .294, 2.121, 0., 0., 0., 15*0., | BIN1.017 |
| X 200*0. / | BIN1.018 |
| END | BIN1.019 |

```

$1BETC BDR.
      BLOCK DATA
C      X IS R**2
C
      COMMON /BLOCKR/ X(20)
      DATA X / 36.548312,
X           39.358308,
X           42.272363,
X           46.838556,
X           48.412651,
X           50.012759,
X           14*0. /
      END

```

```

BDR.0001
BDR.0002
BDR.0003
BDR.0004
BDR.0005
BDR.0006
BDR.0007
BDR.0008
BDR.0009
BDR.0010
BDR.0011
BDR.0012
BDR.0013

```

AFML-TR-67-121
PART III

```

$IRETC BD.      DECK
C
C      BLOCK DATA SUBPROGRAM FOR 122 CO VALUES AND 7 W2 VALUES
C
C      BLOCK DATA
C
C      COMMON /BLOCKW/ W2(20)
C      COMMON /BLOCKC/ CO(160)
C
C      DATA W2
1/416930.0, 583702.0, 817182.8, 1144056., 1601678., 2242349.,
22139289., 13*0. /
C
C      DATA CO
1/.170E-5, .350E-5, .530E-4, .115E-3, .850E-4, .550E-4, .310E-4,
2 .750E-5, .700E-5, .650E-5, .820E-5, .100E-4, .135E-4, .170E-4,
3 .260E-4, .340E-4, .360E-4, .370E-4, .300E-4, .225E-4, .215E-4,
4 .205E-4, .301E-4, .400E-4, .435E-4, .470E-4, .410E-4, .350E-4,
5 .280E-4, .210E-4, .225E-4, .235E-4, .245E-4, .260E-4, .258E-4,
6 .255E-4, .245E-4, .235E-4, .225E-4, .210E-4, .205E-4, .200E-4,
7 .310E-4, .420E-4, .440E-4, .420E-4, .380E-4, .340E-4, .300E-4,
8 .260E-4, .255E-4, .250E-4, .245E-4, .240E-4, .232E-4, .225E-4,
9 .215E-4, .205E-4, .190E-4, .170E-4, .163E-4, .155E-4, .175E-4,
X .190E-4, .225E-4, .260E-4, .240E-4, .220E-4, .195E-4, .190E-4,
1 .182E-4, .175E-4, .170E-4, .165E-4, .160E-4, .150E-4, .140E-4,
2 .130E-4, .110E-4, .900E-5, .650E-5, .450E-5, .500E-5, .550E-5,
3 .820E-5, .110E-4, .113E-4, .115E-4, .920E-5, .700E-5, .600E-5,
4 .500E-5, .400E-5, .300E-5, .230E-5, .150E-5, .100E-5, .150E-5,
5 .220E-5, .300E-5, .400E-5, .500E-5, .470E-5, .450E-5, .420E-5,
6 .400E-5, .500E-5, .600E-5, .680E-5, .750E-5, .700E-5, .650E-5,
7 .580E-5, .500E-5, .420E-5, .350E-5, .300E-5, .200E-5, .150E-5,
8 .100E-5, .500E-6, .0, 0., 0., 36*0. /
      END

```

```

BD.00001
BD.00002
BD.00003
BD.00004
BD.00005
BD.00006
BD.00007
BD.00008
BD.00009
BD.00010
BD.00011
BD.00012
BD.00013
BD.00014
BD.00015
BD.00016
BD.00017
BD.00018
BD.00019
BD.00020
BD.00021
BD.00022
BD.00023
BD.00024
BD.00025
BD.00026
BD.00027
BD.00028
BD.00029
BD.00030
BD.00031
BD.00032
BD.00033

```

```

*BTFC RD.      DECK
C
C   BLOCK DATA SUBPROGRAM FOR 160 CO VALUES AND 7 W2 VALUES
C
C   BLOCK DATA
C
C   COMMON /BLOCKW/ W2(20)
C   COMMON /BLOCKC/ CO(160)
C
C   DATA W2
1/416930.0, 583702.0, 817182.8, 1144056., 1601678., 2242349.,
23139289., 13*0.      /
C
C   DATA (CO(I),I=1,114)
1/.65000E-5,.90000E-5,.11000E-4,.13250E-4,.16000E-4,.19250E-4,
2 .23250E-4,.28000E-4,.33500E-4,.40000E-4,.47750E-4,.56750E-4,
3 .67250E-4,.79500E-4,.93750E-4,.11025E-3,.12925E-3,.15150E-3,
4 .17700E-3,.20600E-3,.23925E-3,.27725E-3,.32050E-3,.36975E-3,
5 .42525E-3,.48775E-3,.55825E-3,.63725E-3,.72575E-3,.82450E-3,
6 .93425E-3,.105575E-2,.119025E-2,.133875E-2,.150175E-2,.168025E-2,BD.00020
7 .18455E-2,.208825E-2,.231925E-2,.256950E-2,.283950E-2,.313025E-2,BD.00021
8 .34420E-2,.377525E-2,.413075E-2,.450825E-2,.490775E-2,.532950E-2,BD.00022
9 .577325E-2,.623825E-2,.672375E-2,.722900E-2,.775275E-2,.829375E-2,BD.00023
X.885050E-2,.942100E-2,.01000325 .,01059475 .,01119325 .,01179625 .,BD.00024
1.01240075 .,01300375 .,01360175 .,01419175 .,01477075 .,01533475 .,BD.00025
2.01588050 .,01640475 .,01690400 .,01737525 .,01781500 .,01822000 .,BD.00026
3.01858775 .,01891575 .,01920150 .,01944275 .,01963800 .,01978575 .,BD.00027
4.01988475 .,01993450 .,01993450 .,01988475 .,01978575 .,01963800 .,BD.00028
5.01944275 .,01920150 .,01891575 .,01858775 .,01822000 .,01781500 .,BD.00029
6.01737525 .,01690400 .,01640475 .,01588050 .,01533475 .,01477075 .,BD.00030
7.01419175 .,01360175 .,01300375 .,01240075 .,01179625 .,01119325 .,BD.00031
8.01059475 .,01000325 .,00942010 .,00885050 .,00829375 .,00775275 .,BD.00032
9.00722900 .,00672375 .,00623825 .,00577325 .,00532950 .,00490775 /BD.00033
C   DATA (CO(I),I=115,160)
1/.00450825,.00413075 .,00377525 .,00344200 .,00313025 .,00283950 .,BD.00035
2.00256950,.00231925 .,00208825 .,00187550 .,00168025 .,00150175 .,BD.00036
3.001338750,.00119025 .,00105575 .,934250E-3,.824500E-3,.725750E-3,BD.00037
4.637250E-3,.558250E-3,.487750E-3,.425250E-3,.369750E-3,.320050E-3,BD.00038
5.277250E-3,.239250E-3,.206000E-3,.177000E-3,.151500E-3,.129250E-3,BD.00039
6.110250E-3,.937500E-4,.795000E-4,.672500E-4,.567500E-4,.477500E-4,BD.00040
7.400000E-4,.335000E-4,.280000E-4,.232500E-4,.192500E-4,.160000E-4,BD.00041
8.132500E-4,.110000E-4,.900000E-5,.650000E-5 /
C   END
C

```

AFML-TR-67-121
PART III

| | | |
|-------------|---|----------|
| \$IBFTC BD. | DECK | BD.00001 |
| C | | BD.00002 |
| C | BLOCK DATA SUBPROGRAM FOR 32 CO VALUES AND 7 W2 VALUES | BD.00003 |
| C | | BD.00004 |
| C | BLOCK DATA | BD.00005 |
| C | | BD.00006 |
| C | COMMON /BLOCKW/ W2(20) | BD.00007 |
| C | COMMON /BLOCKC/ CO(160) | BD.00008 |
| C | | BD.00009 |
| C | DATA W2 | BD.00010 |
| | 1/416930.0, 583702.0, 817182.8, 1144056., 1601678., 2242349., | BD.00011 |
| | 23139289., 13*0. / | BD.00012 |
| C | | BD.00013 |
| | DATA CO | BD.00014 |
| | 1/.9132400E-3, .3044140E-2, .6392700E-2, .1053272E-1, .1497716E-1 | BD.00015 |
| | 2..2051750E-1, .2538812E-1, .3031964E-1, .3494672E-1, .3975646E-1 | BD.00016 |
| | 3..4383562E-1, .4706240E-1, .4968036E-1, .5144596E-1, .5296804E-1 | BD.00017 |
| | 4..5382040E-1, .5461188E-1, .5467276E-1, .5418570E-1, .5333334E-1 | BD.00018 |
| | 5..5144596E-1, .4858448E-1, .4407914E-1, .3835616E-1, .3117200E-1 | BD.00019 |
| | 6..2496194E-1, .1960426E-1, .1503806E-1, .1120244E-1, .7427700E-2 | BD.00020 |
| | 7..4322599E-2, .8140020E-4, 128*0. / | BD.00021 |
| | END | BD.00022 |

AFML-TR-67-121
PART III

| | | |
|---------------|--|----------|
| \$IBMAP RAND. | 100,DECK | RAND.001 |
| * | GENERATES UNIFORM RANDOM NUMBERS | RAND.002 |
| * | R=FLRAN(Y), Y DUMMY GIVES REAL NUMBER | RAND.003 |
| * | CALL SAVE(Z) GIVES LAST OCTAL VALUE | RAND.004 |
| * | CALL VALUE(Z) GIVES STARTING OCTAL VALUE | RAND.005 |
| ENTRY | FLRAN | RAND.006 |
| ENTRY | SAVE | RAND.007 |
| ENTRY | VALUE | RAND.008 |
| FLRAN LDQ | RANDOM | RAND.009 |
| MPY | GENERA | RAND.010 |
| STQ | RANDOM | RAND.011 |
| CLA | AAA | RAND.012 |
| LGL | 28 | RAND.013 |
| FAD | AAA | RAND.014 |
| TRA | 1,4 | RAND.015 |
| VALUE CLA* | 3,4 | RAND.016 |
| STO | RANDOM | RAND.017 |
| TRA | 1,4 | RAND.018 |
| SAVE CLA | RANDOM | RAND.019 |
| STO* | 3,4 | RAND.020 |
| TRA | 1,4 | RAND.021 |
| RANDOM OCT | 343277244615 | RAND.022 |
| AAA OCT | 172000000100 | RAND.023 |
| GENERA OCT | 343277244615 | RAND.024 |
| END | | RAND.025 |

| | |
|--|----------|
| *18ETC PLOTR. DECK | PLOTR.01 |
| SUBROUTINE PLOTR | PLOTR.02 |
| COMMON SCH(4), SCF(4) | PLOTR.03 |
| COMMON /PLTR/ PDATA(438), IPLTS, HEAD(12) | PLOTR.04 |
| COMMON /BINP1/ DH, H, NC, F(162), CM(162), SKIP2 | PLOTR.05 |
| COMMON /BLPS/ N, NM, BF(1000), BM(1000), K | PLOTR.06 |
| COMMON /BIN2/ USEIN2 | PLOTR.07 |
| DATA HTITLE /1HM/, FTITLE /4HF(M)/ | PLOTR.08 |
| LOGICAL SKIP2 | PLOTR.09 |
| LOGICAL USEIN2 | PLOTR.10 |
| IF (SKIP2) RETURN | PLOTR.11 |
| IF (.NOT. USEIN2) GO TO 7 | PLOTR.12 |
| SCH(1) = BM(1) | PLOTR.13 |
| SCH(2) = BM(K) | PLOTR.14 |
| GO TO 8 | PLOTR.15 |
| 7 DM = DH/H | PLOTR.16 |
| DO 5 I = 1,NC | PLOTR.17 |
| 5 F(I) = F(I) / DM | PLOTR.18 |
| SCH(1) = AMIN1(CM(1), BM(1)) | PLOTR.19 |
| SCH(2) = AMAX1(CM(NC), BM(K)) | PLOTR.20 |
| 8 SCF(1) = 0. | PLOTR.21 |
| SCF(2) = 0. | PLOTR.22 |
| IF (USEIN2) GO TO 15 | PLOTR.23 |
| DO 10 I = 1,NC | PLOTR.24 |
| 10 SCF(2) = AMAX1(SCF(2), F(I)) | PLOTR.25 |
| 15 DO 20 I = 1,K | PLOTR.26 |
| 20 SCF(2) = AMAX1(SCF(2), BF(I)) | PLOTR.27 |
| CALL SCALE (SCH, 15., 2, 1, 10.) | PLOTR.28 |
| CALL SCALE (SCF, 10., 2, 1, 10.) | PLOTR.29 |
| IF (USEIN2) GO TO 25 | PLOTR.30 |
| CM(NC+1) = SCH(3) | PLOTR.31 |
| CM(NC+2) = SCH(4) | PLOTR.32 |
| F(NC+1) = SCF(3) | PLOTR.33 |
| F(NC+2) = SCF(4) | PLOTR.34 |
| 25 BM(K+1) = SCH(3) | PLOTR.35 |
| BM(K+2) = SCH(4) | PLOTR.36 |
| BF(K+1) = SCF(3) | PLOTR.37 |
| BF(K+2) = SCF(4) | PLOTR.38 |
| CALL PLOT (5., -11., -3) | PLOTR.39 |
| CALL PLOT (0., .5 , -3) | PLOTR.40 |
| CALL AXIS (0., 0., HTITLE, -1, 16., 0., SCH(3), SCH(4), 10.) | PLOTR.41 |
| CALL AXIS (0., 0., FTITLE, 4, 10., 90., SCF(3), SCF(4), 10.) | PLOTR.42 |
| CALL SYMBOL (1., 9.5, .25, HEAD, 0., 72) | PLOTR.43 |
| IF (.NOT. USEIN2) CALL LINE (CM, F, NC, 1, 0, 0) | PLOTR.44 |
| CALL LINE (BM, BF, K, 1, 1, 1) | PLOTR.45 |
| CALL PLOT (15., 0., -3) | PLOTR.46 |
| IPLTS = IPLTS + 1 | PLOTR.47 |
| RETURN | PLOTR.48 |
| END | PLOTR.49 |

| | |
|--|----------|
| SUBROUTINE ORDER. DECK | ORDER.01 |
| SUBROUTINE ORDER (X, Y, NM, N, K) | ORDER.02 |
| COMMON /PRTCTL/ WRITE | ORDER.03 |
| DIMENSION X(1), Y(1) | ORDER.04 |
| LOGICAL WRITE | ORDER.05 |
| J = 0 | ORDER.06 |
| DO 10 LN = 1,N | ORDER.07 |
| N20 = (LN-1) * 20 | ORDER.08 |
| I = N20 + 1 | ORDER.09 |
| J = J + 1 | ORDER.10 |
| X(J) = X(L) | ORDER.11 |
| Y(J) = Y(L) | ORDER.12 |
| DO 10 I = 2,NM | ORDER.13 |
| L = N20 + I | ORDER.14 |
| J = J + 1 | ORDER.15 |
| X(J) = X(L) | ORDER.16 |
| 10 Y(J) = Y(L) | ORDER.17 |
| K = J | ORDER.18 |
| 20 TEST = 0. | ORDER.19 |
| DO 30 I = 2,J | ORDER.20 |
| IF (X(I-1) .LE. X(I)) GO TO 30 | ORDER.21 |
| XS = X(I-1) | ORDER.22 |
| YS = Y(I-1) | ORDER.23 |
| X(I-1) = X(I) | ORDER.24 |
| Y(I-1) = Y(I) | ORDER.25 |
| X(I) = XS | ORDER.26 |
| Y(I) = YS | ORDER.27 |
| TEST = 1. | ORDER.28 |
| 30 CONTINUE | ORDER.29 |
| IF ((TEST .EQ. 0.) .OR. (J .EQ. 2)) GO TO 40 | ORDER.30 |
| J = J - 1 | ORDER.31 |
| GO TO 20 | ORDER.32 |
| 40 IF (WRITE). WRITE (6,600) (I, X(I), Y(I), I=1,K) | ORDER.33 |
| RETURN | ORDER.34 |
| 600 FORMAT (1H1, 15X, 1HM, 19X, 1HF / 1H / | ORDER.35 |
| X (1X, 13, 2E20.7)) | ORDER.36 |
| END | ORDER.37 |


```

$IBFTC DETA.  DECK
SUBROUTINE DETA ( A, B, NM, NROW )
COMMON /PRCTL/ WRITE
DIMENSION A(51,120), B(NM,NM)
LOGICAL WRITE
IF ( WRITE ) WRITE (6,600)
J = 1
10 DO 20 I = 1,NM
DO 20 K = 1,NM
JI = J + I
20 B(I,K) = A(JI,K)
D = DET ( B, NM )
IF ( WRITE ) WRITE (6,601) J, D
J = J + NM
IF ( (J+NM-1) .GT. NROW ) RETURN
GO TO 10
600 FORMAT ( 13HODETERMINANTS )
601 FORMAT ( 3X, I4, E19.7 )
END

```

```

DETA.001
DETA.002
DETA.003
DETA.004
DETA.005
DETA.006
DETA.007
DETA.008
DETA.009
DETA.010
DETA.011
DETA.012
DETA.013
DETA.014
DETA.015
DETA.016
DETA.017
DETA.018
DETA.019

```

```

$IBFTC DLETE. DECK
SUBROUTINE DLETE ( IRHS, NROW, NRP1, NCOL, NM, * )
COMMON JRHS(100)
COMMON /BIN1/ NW, NX, XL(2), Z(20), U(20,20)
COMMON /DLT/ NWW, NXX
COMMON /PRTCTL/ WRITE
LOGICAL WRITE
KRHS = 2 * IRHS
IF ( KRHS .GT. 100 ) GO TO 900
READ (5,500) (JRHS(I),I=1,KRHS)
IF ( WRITE ) WRITE (6,601) IRHS, (JRHS(I),I=1,KRHS)
DO 10 I = 1,KRHS,2
K = JRHS(I)
L = JRHS(I+1)
10 U(K,L) = 0.
NWW = NW
I = 0
DO 30 K = 1,NW
USUM = 0.
DO 20 L = 1,NX
20 USUM = USUM + U(K,L)
IF ( USUM .EQ. 0. ) GO TO 28
I = I + 1
DO 25 J = 1,NX
25 U(I,J) = U(K,J)
XL(I) = XL(K)
GO TO 30
28 NWW = NWW - 1
30 CONTINUE
NXX = NX
J = 0
DO 50 L = 1,NX
USUM = 0.
DO 40 K = 1,NWW
40 USUM = USUM + U(K,L)
IF ( USUM .EQ. 0. ) GO TO 48
J = J + 1
DO 45 I = 1,NWW
45 U(I,J) = U(I,L)
Z(J) = Z(L)
GO TO 50
48 NXX = NXX - 1
50 CONTINUE
NROW = NXX * NWW
NRP1 = NROW + 1
NCOL = NM + (2*NROW)
IF ( .NOT. WRITE ) RETURN
WRITE (6,602) NWW, NXX
DO 60 I = 1,NWW
60 WRITE (6,603) I, (U(I,J),J=1,NXX)
RETURN
900 WRITE (6,600) IRHS
RETURN 1
500 FORMAT ( 27( 2I1,1X) )
600 FORMAT ( 45HNUMBER OF U MATRIX DELETIONS GREATER THAN 50 /
X
7H0IRHS = I4 )
601 FORMAT ( 1HA, I3, 41H ELEMENTS OF MATRIX -U- HAVE BEEN DELETED /
X
33(2X,2I1) )
602 FORMAT ( 20HAU MATRIX (ADJUSTED), I10, 5H ROWS, I6, 8H COLUMNS )
603 FORMAT ( 4HOROW, I4, 1X, 6E20.7 / (9X, 6E20.7) )
END

```

DLETE001
DLETE002
DLETE003
DLETE004
DLETE005
DLETE006
DLETE007
DLTF008
DLTF009
DLETE010
DLETE011
DLETE012
DLETE013
DLETE014
DLETE015
DLETE016
DLETE017
DLTF018
DLETE019
DLTF020
DLETE021
DLETE022
DLETE023
DLETE024
DLETE025
DLETE026
DLETE027
DLETE028
DLETE029
DLETE030
DLETE031
DLETE032
DLETE033
DLTF034
DLETE035
DLETE036
DLETE037
DLETE038
DLETE039
DLETE040
DLETE041
DLETE042
DLTF043
DLETE044
DLETE045
DLETE046
DLETE047
DLETE048
DLETE049
DLETE050
DLETE051
DLTF052
DLETE053
DLETE054
DLETE055
DLETE056
DLETE057
DLETE058
DLETE059
DLETE060
DLETE061

| | |
|--|----------|
| \$IBFIC LPS. DECK | LPS.0001 |
| SUBROUTINE LPS (*) | LPS.0002 |
| COMMON IE(20), E(51,51), X(51), P(51), Y(51), JH(51), | LPS.0003 |
| X KB(120), KOUT(7), ERR(8) | LPS.0004 |
| COMMON /BIN1/ NW, NX, XL(20), Z(20), U(20,20) | LPS.0005 |
| COMMON /BLPS/ N, NM, BF(20,50), BM(20,50), K | LPS.0006 |
| COMMON /DLT/ NWW, NXX | LPS.0007 |
| COMMON /PRTCTL/ WRITE | LPS.0008 |
| DIMENSION A(51,120), B(51), INFIX(8), TOL(4), BA(400) | LPS.0009 |
| DIMENSION ARR(50), BS(51) | LPS.0010 |
| LOGICAL WRITE | LPS.0011 |
| DATA NMAX/50/, NMMAX/20/, NCOLMX/120/, NROWMX/50/ | LPS.0012 |
| NAME LIST /LP/ XM11, G, NM, N, IRHS, PERT | LPS.0013 |
| PERT = 0. | LPS.0014 |
| IRHS = 0 | LPS.0015 |
| READ (5,LP) | LPS.0016 |
| IF (WRITE) WRITE (6,600) XM11, G, NM, N | LPS.0017 |
| IF (N .GT. NMAX) GO TO 900 | LPS.0018 |
| IF (NM .GT. NMMAX) GO TO 904 | LPS.0019 |
| NROW = NW * NX | LPS.0020 |
| IF (NROW .GT. NROWMX) GO TO 902 | LPS.0021 |
| NCOL = NM + (2*NROW) | LPS.0022 |
| IF (NCOL .GT. NCOLMX) GO TO 901 | LPS.0023 |
| NRPI = NROW + 1 | LPS.0024 |
| NWW = NW | LPS.0025 |
| NXX = NX | LPS.0026 |
| IF (IRHS .GT. 0) | LPS.0027 |
| X CALL DLETE (IRHS, NROW, NRPI, NCOL, NM, \$903) | LPS.0028 |
| 20 DO 30 I = 1, NRPI | LPS.0029 |
| DO 30 J = 1, NCOL | LPS.0030 |
| 30 A(I,J) = 0. | LPS.0031 |
| DO 35 I = 1, NMMAX | LPS.0032 |
| DO 35 J = 1, NMAX | LPS.0033 |
| 35 BF(I,J) = 0. | LPS.0034 |
| INFIX(1) = 4 | LPS.0035 |
| INFIX(2) = NCOL | LPS.0036 |
| INFIX(3) = 51 | LPS.0037 |
| INFIX(4) = NRPI | LPS.0038 |
| INFIX(5) = 2 | LPS.0039 |
| INFIX(6) = 1 | LPS.0040 |
| INFIX(7) = 500 | LPS.0041 |
| INFIX(8) = 20 | LPS.0042 |
| TOL(1) = 1.E-7 | LPS.0043 |
| TOL(2) = 1.E-7 | LPS.0044 |
| TOL(3) = 1.E-6 | LPS.0045 |
| TOL(4) = 1.E-10 | LPS.0046 |
| PRM = 0. | LPS.0047 |
| R(1) = 0. | LPS.0048 |
| DO 50 I = 1, NROW | LPS.0049 |
| IN = (I-1) / NXX | LPS.0050 |
| K = IN + 1 | LPS.0051 |
| L = I - (IN*NXX) | LPS.0052 |
| R(I+1) = U(K,L) | LPS.0053 |
| IF (PERT .GT. 0.) B(I+1) = B(I+1)*(1.+2.*PERT*(FLRAN(X)-.5)) | LPS.0054 |
| 50 BS(I+1) = B(I+1) | LPS.0055 |
| LN = 1 | LPS.0056 |
| SQRTG = SQRT(G) | LPS.0057 |
| GMG = SQRTG - (1. / SQRTG) | LPS.0058 |
| XN = 1. / FLOAT(N) | LPS.0059 |
| BM(1,1) = XM11 | LPS.0060 |
| NROWS = NROW | LPS.0061 |

AFML-TR-67-121
PART III

| | |
|--|----------|
| 55 IF (WRITE) WRITE (6,612) LN, N | LPS.0062 |
| NROW = NROWS | LPS.0063 |
| NRPI = NROW + 1 | LPS.0064 |
| DO 60 J = 1,NM | LPS.0065 |
| RM(J,LN) = RM(1,LN)* G**(J-1) | LPS.0066 |
| DBM = GMG * RM(J,LN) | LPS.0067 |
| DO 60 I = 1,NROW | LPS.0068 |
| IN = (I-1) / NXX | LPS.0069 |
| K = IN +1 | LPS.0070 |
| L = I- (IN*NXX) | LPS.0071 |
| XLKM = XL(K) * BM(J,LN) | LPS.0072 |
| XNUM = XLKM**2 * EXP(-XLKM * Z(L)) | LPS.0073 |
| DENOM = 1. - EXP(-XLKM) | LPS.0074 |
| 60 A(I+1,J) = (XNUM / DENOM) * DBM | LPS.0075 |
| II = 1 | LPS.0076 |
| DO 64 I = 2,NRPI | LPS.0077 |
| B(I) = RS(I) | LPS.0078 |
| IF (B(I) .EQ. 0.) GO TO 64 | LPS.0079 |
| II = II + 1 | LPS.0080 |
| B(II) = B(I) | LPS.0081 |
| DO 62 J = 1,NM | LPS.0082 |
| 62 A(II,J) = A(I,J) | LPS.0083 |
| 64 CONTINUE | LPS.0084 |
| NRPI = II | LPS.0085 |
| NROW = NRPI - 1 | LPS.0086 |
| INFIX(4) = NRPI | LPS.0087 |
| J = NM | LPS.0088 |
| DO 68 I = 2,NRPI | LPS.0089 |
| J = J + 1 | LPS.0090 |
| A(1,J) = 1.0 | LPS.0091 |
| A(1,J) = 1.0 | LPS.0092 |
| J = J + 1 | LPS.0093 |
| A(1,J) = 1.0 | LPS.0094 |
| 68 A(1,J) = -1.0 | LPS.0095 |
| CALL DETA (A, BA, NM, NROW) | LPS.0096 |
| IF (WRITE) WRITE (6,601) (M,M=1,30) | LPS.0097 |
| DO 80 I = 2,NRPI | LPS.0098 |
| DO 70 J = 1,NM | LPS.0099 |
| IF(J) = -99 | LPS.0100 |
| 70 IF (A(I,J) .NE. 0.) IF(J) = ALOG10(A(I,J)) | LPS.0101 |
| IM1 = I - 1 | LPS.0102 |
| 80 IF (WRITE) WRITE (6,602) IM1, (IF(J),J=1,NM) | LPS.0103 |
| CALL SIMPLX (INFIX, A, B, TOL, PRM, KOUT, ERR, JH, X, P, Y,KB,E) | LPS.0104 |
| IF (WRITE) WRITE (6,603) | LPS.0105 |
| DO 90 J = 1,NRPI | LPS.0106 |
| MX = JH(J) | LPS.0107 |
| 90 IF ((MX .GT. 0) .AND. (MX .LE. NM)) BF(MX,LN) = X(J) | LPS.0108 |
| ABR(LN) = 0. | LPS.0109 |
| DO 110 I = 2,NRPI | LPS.0110 |
| BC = 0. | LPS.0111 |
| DO 100 J = 1,NM | LPS.0112 |
| 100 BC = BC + BF(J,LN) * A(I,J) | LPS.0113 |
| BD = B(I) - BC | LPS.0114 |
| AB = ABS(BD) / B(I) | LPS.0115 |
| IF (WRITE) WRITE (6,604) B(I), BC, BD, AB | LPS.0116 |
| 110 ABR(LN) = ABR(LN) + AB | LPS.0117 |
| ABR(LN) = ABR(LN) / FLOAT(NROW) | LPS.0118 |
| IF (.NOT. WRITE) GO TO 115 | LPS.0119 |
| WRITE (6,605) ABR(LN) | LPS.0120 |
| WRITE (6,606) LN, (BF(M,LN),M=1,NM) | LPS.0121 |
| WRITE (6,610) | LPS.0122 |

AFML-TR-67-121
PART III

```

115 LN = LN + 1                                LPS.0123
    IF ( LN .GT. N ) GO TO 120                  LPS.0124
    RM(1,LN) = RM(1,LN-1) * G**XN              LPS.0125
    GO TO 55                                    LPS.0126
120 IF ( WRITE ) WRITE (6,610)                  LPS.0127
    WRITE (6,615)                               LPS.0128
    ARBAVG = 0.                                LPS.0129
    DO 130 I = 1,N                              LPS.0130
        ARBAVG = ARBAVG + ABR(I)                LPS.0131
130 WRITE (6,614) I, ABR(I), (BF(M,I),M=1,NM)   LPS.0132
    ARBAVG = ARBAVG / FLOAT( N )                LPS.0133
    WRITE (6,613) ARBAVG                       LPS.0134
    CALL ORDER ( RM, BF, NM, N, K )             LPS.0135
    RETURN                                       LPS.0136
900 WRITE (6,607) N                            LPS.0137
    RETURN 1                                    LPS.0138
901 WRITE (6,608) NCOL                         LPS.0139
    RETURN 1                                    LPS.0140
902 WRITE (6,609) NROW                         LPS.0141
903 RETURN 1                                    LPS.0142
904 WRITE (6,611) NM                           LPS.0143
    RETURN 1                                    LPS.0144
600 FORMAT ( 33HAFIRST MOLECULAR WEIGHT        = E16.7 /
X          33H MOLECULAR WEIGHT MULTIPLIER    = E16.7 /
X          33H NUMBER OF MOLECULAR WT. VALUES = I8 /
X          33H NUMBER OF LP SETS FOR SOLUTION = I8 ) LPS.0145
601 FORMAT ( 1HA, 57X, 14HA MATRIX (LOG) / 1HC, 60X, 7HCOLUMNS /
X          5H ROWS, 30I4 / 1H )                LPS.0146
602 FORMAT ( 1X, I3, 1X, 30I4 / ( 5X, 30I4 ) ) LPS.0147
603 FORMAT ( 1H1, 7X, 9HINPUT RHS, 7X, 12HCOMPUTED RHS, 6X,
X          14HRHS DIFFERENCE, 4X, 16HABS REL DIFF RHS / 1H ) LPS.0148
604 FORMAT ( 1X, 4E18.7 )                       LPS.0149
605 FORMAT ( 34HOAVERAGE RELATIVE DIFFERENCE RHS = E16.7 ) LPS.0150
606 FORMAT ( 9HOSOLUTION, I4, 7X, 7E16.7 / (20X, 7E16.7) ) LPS.0151
607 FORMAT ( 4HON = I4, 42H IS GREATER THAN DIMENSION FOR NO. OF SETS ) LPS.0152
608 FORMAT ( 56HONUMBER OF COLUMNS FOR -A- MATRIX GREATER THAN DIMENSION
XON / 7HONROW = I4 ) LPS.0153
609 FORMAT ( 53HONUMBER OF ROWS FOR -A- MATRIX GREATER THAN DIMENSION
X / 7HOROW = I4 ) LPS.0154
610 FORMAT ( 1H1 )                               LPS.0155
611 FORMAT ( 5HONM = I4, 59H IS GREATER THAN DIMENSION FOR NUMBER OF
XOLUTIONS PER SET ) LPS.0156
612 FORMAT ( 1H0, 57X, 3HSET I3, 3H OF I3 )      LPS.0157
613 FORMAT ( 1H0 / 20X, 38HTHE AVERAGE REL. DIFF. FOR ALL SETS IS
X          E16.7 ) LPS.0158
614 FORMAT ( 1H0, I4, E22.7, 9X, 6E16.7 / (36X, 6E16.7) ) LPS.0159
615 FORMAT (6H0 SFT,6X,15HAVG. REL. ERROR,12X,11HSOLUTIONS ,82(1H*)
X / 1H ) LPS.0160
    END                                         LPS.0161

```

AFML-TR-67-121
PART III

| | |
|---|----------|
| *IRFIC IN1. DECK | IN1.0001 |
| SUBROUTINE IN1 (*) | IN1.0002 |
| COMMON /BLOCKR/ X(20) | IN1.0003 |
| COMMON /BLOCKC/ C0(160) | IN1.0004 |
| COMMON /BLOCKW/ W2(20) | IN1.0005 |
| COMMON /BIN1/ NW, NX, XL(20), Z(20), U(20,20) | IN1.0006 |
| COMMON /BIN2/ USEIN2 | IN1.0007 |
| COMMON /BINP1/ DH, H, NC, F(162), H0(162), SKIP2 | IN1.0008 |
| COMMON /PRTCTL/ WRITE | IN1.0009 |
| C DATA NXMAX/20/, NHMAX/160/, NWMAX/20/ | IN1.0010 |
| C LOGICAL WRITE, SKIP1, SKIP2 | IN1.0011 |
| C LOGICAL USEIN2 | IN1.0012 |
| C NAME LIST /INP1/ NW, NX, H1, DH, NH, H, CZ, X1, DX, | IN1.0013 |
| X IST, LST, LAST, WRITE, SKIP1, SKIP2, USEIN2 | IN1.0014 |
| C WRITE = .FALSE. | IN1.0015 |
| SKIP1 = .FALSE. | IN1.0016 |
| SKIP2 = .FALSE. | IN1.0017 |
| USEIN2 = .FALSE. | IN1.0018 |
| LAST = 1 | IN1.0019 |
| READ (5,INP1) | IN1.0020 |
| IF (LAST .EQ. 0) RETURN 1 | IN1.0021 |
| IF (SKIP1) GO TO 75 | IN1.0022 |
| IF (NX .GT. NXMAX) GO TO 900 | IN1.0023 |
| IF (NH .GT. NHMAX) GO TO 901 | IN1.0024 |
| IF (NW .GT. NWMAX) GO TO 902 | IN1.0025 |
| IF (USEIN2) CALL IN2 (\$85) | IN1.0026 |
| NC = LST - IST + 1 | IN1.0027 |
| IF (NC .GT. NHMAX) GO TO 903 | IN1.0028 |
| IF (X1 .LE. 0.) GO TO 5 | IN1.0029 |
| X(1) = X1 | IN1.0030 |
| DO 10 L = 2,NX | IN1.0031 |
| 10 X(L) = X1 + FLOAT(L-1) * DX | IN1.0032 |
| 5 H0(1) = H1 | IN1.0033 |
| H0(1) = H1 | IN1.0034 |
| DO 20 N = 2,NH | IN1.0035 |
| 20 H0(N) = H1 + FLOAT(N-1) * DH | IN1.0036 |
| DO 30 I = 1,NC | IN1.0037 |
| IC = I + IST - 1 | IN1.0038 |
| F(I) = C0(IC) / CZ | IN1.0039 |
| 30 H0(I) = H0(IC) / H | IN1.0040 |
| B = X(NX) | IN1.0041 |
| BMM = B - X1 | IN1.0042 |
| DO 40 K = 1,NW | IN1.0043 |
| 40 XL(K) = BMM * H * W2(K) | IN1.0044 |
| DO 50 L = 1,NX | IN1.0045 |
| 50 Z(L) = (B - X(L)) / BMM | IN1.0046 |
| DO 70 K = 1,NW | IN1.0047 |
| DO 70 L = 1,NX | IN1.0048 |
| U(K,L) = 0. | IN1.0049 |
| DO 70 N = 1,NC | IN1.0050 |
| XLM = XL(K) * H0(N) | IN1.0051 |
| XNUM = XLM**2 * EXP(-XLM*Z(L)) | IN1.0052 |
| DENOM = 1. - EXP(-XLM) | IN1.0053 |
| 70 U(K,L) = U(K,L) + (XNUM / DENOM) * F(N) | IN1.0054 |
| 75 IF (.NOT. WRITE) GO TO 85 | IN1.0055 |
| WRITE (6,601) NW, (W2(K),K=1,NW) | IN1.0056 |
| WRITE (6,602) NX, (Z(L),L=1,NX) | IN1.0057 |
| | IN1.0058 |
| | IN1.0059 |
| | IN1.0060 |
| | IN1.0061 |

AFML-TR-67-121
PART III

| | |
|--|----------|
| WRITE (6,603) NW, NX | IN1.0062 |
| DO 80 K = 1,NW | IN1.0063 |
| 80 WRITE (6,604) K, (U(K,L),L=1,NX) | IN1.0064 |
| 85 RETURN | IN1.0065 |
| 900 WRITE (6,605) NX | IN1.0066 |
| STOP | IN1.0067 |
| 901 WRITE (6,606) NH | IN1.0068 |
| STOP | IN1.0069 |
| 902 WRITE (6,607) NW | IN1.0070 |
| STOP | IN1.0071 |
| 903 WRITE (6,608) NC | IN1.0072 |
| STOP | IN1.0073 |
| 601 FORMAT (1HA, I4, 40H VALUES OF ANGULAR VELOCITY SQUARED - W2 / | IN1.0074 |
| X 1H / (9X, 6E20.7)) | IN1.0075 |
| 602 FORMAT (1H0, I4, 71H VALUES OF THE FUNCTION OF DISTANCE SQUARED F | IN1.0076 |
| XROM CENTER OF ROTATION - Z / 1H / (9X, 6E20.7)) | IN1.0077 |
| 603 FORMAT (9H0U MATRIX, I10, 5H ROWS, I6, 8H COLUMNS) | IN1.0078 |
| 604 FORMAT (4HOROW, I4, 1X, 6E20.7, / (9X, 6E20.7)) | IN1.0079 |
| 605 FORMAT (5HONX = I4, 33H IS GREATER THAN DIMENSION FOR X.) | IN1.0080 |
| 606 FORMAT (5HONH = I4, 33H IS GREATER THAN DIMENSION FOR H.) | IN1.0081 |
| 607 FORMAT (5HONW = I4, 37H IS GREATER THAN DIMENSION FOR OMEGA.) | IN1.0082 |
| 608 FORMAT (5HONC = I4, 41H IS GREATER THAN DIMENSION FOR SELECTED H) | IN1.0083 |
| END | IN1.0084 |

AFML-TR-67-121
PART III

| | |
|--|----------|
| \$IBFTC IN2. DECK | IN2.0001 |
| SUBROUTINE IN2 (*) | IN2.0002 |
| SCRATCH STORAGE | IN2.0003 |
| COMMON DCDX(20,20) | IN2.0004 |
| DIMENSION DNDR(20,20) | IN2.0005 |
| LOGICAL WRITE | IN2.0006 |
| COMMON /BIN1 / NW, NR, XL(20), Z(20), U(20,20) | IN2.0007 |
| COMMON /BLOCKW/ W2(20) | IN2.0008 |
| COMMON /BLOCKR/ X(20) | IN2.0009 |
| COMMON /PRTCTL/ WRITE | IN2.0010 |
| DATA NRMAX/20/, NWMAX/20/ | IN2.0011 |
| NAME LIST /INP2/ NW, NR, H, CO, R1, DR, DCDN, DNDR, W2 | IN2.0012 |
| READ (5,INP2) | IN2.0013 |
| IF (NR .GT. NRMAX) GO TO 900 | IN2.0014 |
| IF (NW .GT. NWMAX) GO TO 901 | IN2.0015 |
| DO 10 L = 1,NR | IN2.0016 |
| IF (R1 .LE. 0.) GO TO 5 | IN2.0017 |
| X(L) = (R1 + FLOAT(L-1) * DR)**2 | IN2.0018 |
| 5 DO 10 K = 1,NW | IN2.0019 |
| 10 DCDX(K,L) = 1.0 / SQRT(X(L)) * DCDN * DNDR(K,L) | IN2.0020 |
| XM = X(1) | IN2.0021 |
| R = X(NR) | IN2.0022 |
| DO 20 K = 1, NW | IN2.0023 |
| 20 XL(K) = (B-XM) * H * W2(K) | IN2.0024 |
| DO 30 L = 1, NR | IN2.0025 |
| 30 Z(L) = (B-X(L)) / (B-XM) | IN2.0026 |
| DO 40 L = 1,NR | IN2.0027 |
| DO 40 K = 1,NW | IN2.0028 |
| 40 U(K,L) = ((B-XM) / CO) * DCDX(K,L) | IN2.0029 |
| IF (.NOT. WRITE) GO TO 60 | IN2.0030 |
| WRITE (6,601) NW, (W2(K),K=1,NW) | IN2.0031 |
| WRITE (6,602) NR, (Z(L),L=1,NR) | IN2.0032 |
| WRITE (6,603) NW, NR | IN2.0033 |
| DO 50 K = 1,NW | IN2.0034 |
| 50 WRITE (6,604) K, (U(K,L),L=1,NR) | IN2.0035 |
| 60 RETURN 1 | IN2.0036 |
| 900 WRITE (6,605) NR, NRMAX | IN2.0037 |
| STOP | IN2.0038 |
| 901 WRITE (6,606) NW, NWMAX | IN2.0039 |
| STOP | IN2.0040 |
| 601 FORMAT (1HA, 14, 40H VALUES OF ANGULAR VELOCITY SQUARED - W2 / | IN2.0041 |
| X 1H / (9X, 6E20.7)) | IN2.0042 |
| 602 FORMAT (1H0, 14, 71H VALUES OF THE FUNCTION OF DISTANCE SQUARED FIN | IN2.0043 |
| XROM CENTER OF ROTATION - Z / 1H / (9X, 6E20.7)) | IN2.0044 |
| | IN2.0045 |
| | IN2.0046 |
| | IN2.0047 |
| | IN2.0048 |
| | IN2.0049 |
| | IN2.0050 |
| | IN2.0051 |
| | IN2.0052 |
| | IN2.0053 |
| | IN2.0054 |
| | IN2.0055 |
| | IN2.0056 |
| | IN2.0057 |
| | IN2.0058 |
| | IN2.0059 |
| | IN2.0060 |
| | IN2.0061 |

AFML-TR-67-121
PART III

```
603 FORMAT ( 9H0U MATRIX, I10, 5H ROWS, I6, 8H COLUMNS )      IN2.0062
604 FORMAT ( 4H0ROW, I4, 1X, 6E20.7 / ( 9X, 6E20.7 ) )      IN2.0063
605 FORMAT ( 5H0NR = I4, 33H IS GREATER THAN DIMENSION FOR X(,I2,2H).) IN2.0064
606 FORMAT ( 5H0NW = I4, 33H IS GREATER THAN DIMENSION FOR W(,I2,2H).) IN2.0065
C                                                                IN2.0066
      END                                                        IN2.0067
```

| | | |
|--|------------|----------|
| \$IBFTC DET. DECK | L. B. FALL | DET.0001 |
| FUNCTION DET(A,N) | | DET.0002 |
| C | | DET.0003 |
| C DETERMINANT EVALUATING FUNCTION | | DET.0004 |
| C | | DET.0005 |
| C FUNCTION DET(A,N) COMPUTES THE DETERMINANT | | DET.0006 |
| C OF THE N-TH ORDER MATRIX A, WHICH MUST BE | | DET.0007 |
| C DIMENSIONED A(N,N). THE ORIGINAL MATRIX A | | DET.0008 |
| C IS NOT ALTERED. | | DET.0009 |
| C | | DET.0010 |
| C 612 CELLS OF BLANK COMMON ARE USED | | DET.0011 |
| C | | DET.0012 |
| C TO CHANGE DIMENSIONS, CHANGE DIMENSIONS OF ARRAYS B AND PIV, | | DET.0013 |
| C AND ALSO CHANGE VALUE OF NMAX IN THE DATA STATEMENT. | | DET.0014 |
| C | | DET.0015 |
| C DIMENSION A(N,N) | | DET.0016 |
| C COMMON D, I, II, J, K, KCT, KFROM, KTO, NN, NPR, RLE, TPE | | DET.0017 |
| C COMMON B(24,24), PIV(24) | | DET.0018 |
| C DATA NMAX/ 24/ | | DET.0019 |
| C | | DET.0020 |
| C TEST ARGUMENT N TO PREVENT OVERFLOWING BLANK COMMON | | DET.0021 |
| C | | DET.0022 |
| C NN=N | | DET.0023 |
| C IF (NN .GT. NMAX .OR. NN .LE. 0) GO TO 100 | | DET.0024 |
| C | | DET.0025 |
| C MOVE INPUT MATRIX A TO SCRATCH MATRIX B | | DET.0026 |
| C | | DET.0027 |
| C DO 10 I=1,NN | | DET.0028 |
| C DO 10 J=1,NN | | DET.0029 |
| 10 B(I,J)=A(I,J) | | DET.0030 |
| C | | DET.0031 |
| C INITIALIZE DETERMINANT VALUE AND ROW INTERCHANGE COUNT | | DET.0032 |
| C | | DET.0033 |
| C D=1.0 | | DET.0034 |
| C KCT=0 | | DET.0035 |
| C | | DET.0036 |
| C PERFORM ELIMINATION ON N COLUMNS | | DET.0037 |
| C | | DET.0038 |
| C DO 90 I=1,NN | | DET.0039 |
| C | | DET.0040 |
| C SEARCH I-TH SUB-COLUMN FOR I-TH PIVOT ELEMENT | | DET.0041 |
| C | | DET.0042 |
| C TPE=0. | | DET.0043 |
| C DO 30 II=I,NN | | DET.0044 |
| C IF (ARS(B(II,I))-TPE) 30,20,20 | | DET.0045 |
| 20 NPR=II | | DET.0046 |
| C TPE=ARS(B(II,I)) | | DET.0047 |
| 30 CONTINUE | | DET.0048 |
| C | | DET.0049 |
| C IF PIVOT ELEMENT IS ZERO, THEN DET(A,N)=0.0 | | DET.0050 |
| C | | DET.0051 |
| C IF (B(NPR,I)) 35,32,35 | | DET.0052 |
| 32 D=0. | | DET.0053 |
| C GO TO 95 | | DET.0054 |
| C | | DET.0055 |
| C DIVIDE PIVOT ROW BY PIVOT ELEMENT | | DET.0056 |
| C | | DET.0057 |
| 35 DO 40 J=I,NN | | DET.0058 |
| 40 PIV(J)=B(NPR,J)/B(NPR,I) | | DET.0059 |
| C | | DET.0060 |
| C UPDATE THE PRODUCT OF PIVOT ELEMENTS AND SUM OF ROW INTERCHANGES | | DET.0061 |

AFML-TR-67-121
PART III

| | | |
|------|--|----------|
| C | D=D*B(NPR,I) | DET.0062 |
| | KCT=KCT+(NPR-I) | DET.0063 |
| C | | DET.0064 |
| C | ELIMINATE REMAINING ELEMENTS IN I-TH SUB-COLUMN | DET.0065 |
| C | | DET.0066 |
| | KTO=NN | DET.0067 |
| | KFROM=NN | DET.0068 |
| | DO 90 K=I,NN | DET.0069 |
| | IF (KFROM-NPR) 70,80,70 | DET.0070 |
| 70 | RLE=-B(KFROM,I) | DET.0071 |
| | DO 75 J=I,NN | DET.0072 |
| 75 | B(KTO,J)=B(KFROM,J)+RLE*PIV(J) | DET.0073 |
| | KTO=KTO-1 | DET.0074 |
| 80 | KFROM=KFROM-1 | DET.0075 |
| 90 | CONTINUE | DET.0076 |
| | | DET.0077 |
| C | | DET.0078 |
| C | IF TOTAL NO. OF ROW INTERCHANGES WAS ODD, THEN | DET.0079 |
| C | NEGATE THE PRODUCT OF THE PIVOT ELEMENTS | DET.0080 |
| C | | DET.0081 |
| 95 | IF (KCT .NE. 2*(KCT/2)) D=-D | DET.0082 |
| | DET=D | DET.0083 |
| | RETURN | DET.0084 |
| C | | DET.0085 |
| C | GIVE ERROR MESSAGE FOR INCORRECT VALUE OF N | DET.0086 |
| C | AND RETURN TO SYSTEM VIA FXEM | DET.0087 |
| C | | DET.0088 |
| 100 | WRITE (6,1000) NN | DET.0089 |
| | CALL FXEM | DET.0090 |
| | RETURN | DET.0091 |
| C | | DET.0092 |
| 1000 | FORMAT (3HON=,I12,30H IS INCORRECT FOR FUNCTION DET) | DET.0093 |
| | END | DET.0094 |

AFML-TR-67-121
PART III

| | | |
|--------------|--|----------|
| \$IBFTC MSUB | DECK | MSUB.001 |
| CMSUBJ | VERSION 1 OF RS MSUB | MSUB.002 |
| | SUBROUTINE SIMPLX (INFIX,A,B,TOL,PRM,KOUT,ERR,JH,X,P,Y,KB,E) | MSUB.003 |
| C | | MSUB.004 |
| | DIMENSION INFIX(8),A(1),B(1),TOL(4),KOUT(7),ERR(8),JH(1),X(1), | MSUB.005 |
| | 1 P(1),Y(1),KB(1),F(1),ZZ(3), IOFIX(16) , TFRR(8) | MSUB.006 |
| C | | MSUB.007 |
| | EQUIVALENCE (INFLAG,IOFIX(1)), (N , IOFIX(2)) , | MSUB.008 |
| | 1 (ME,IOFIX(3)) , (M,IOFIX(4)) , (MF,IOFIX(5)) , | MSUB.009 |
| | 2 (MC, IOFIX(6)) , (NCUI, IOFIX(7)) , (NVER, IOFIX(8)) , | MSUB.010 |
| | 3 (K, IOFIX(9)) , (IIER, IOFIX(10)) , (INvC , IOFIX(11)) , | MSUB.011 |
| | 4 (NUMVR, IOFIX(12)) , (Numpv, IOFIX(13)) , | MSUB.012 |
| | 5 (INFS, IOFIX(14)) , (JT, IOFIX(15)) , (LA , IOFIX(16)) , | MSUB.013 |
| | 6 (ZZ(1),TPIV), (ZZ(2),TZERO), (ZZ(3),TCOSI) | MSUB.014 |
| C | | MSUB.015 |
| C | MOVE INPUTS ... ZERO OUTPUTS | MSUB.016 |
| | DO 1340 I= 1, 8 | MSUB.017 |
| | TERR(I) = 0.0 | MSUB.018 |
| | IOFIX(I+8) = 0 | MSUB.019 |
| 1340 | IOFIX(I) = INFIX(I) | MSUB.020 |
| | LA = 0 | MSUB.021 |
| | DO 1308 I = 1 , 3 | MSUB.022 |
| 1308 | ZZ(I) = TOL(I) | MSUB.023 |
| | TCOST = - ABS (TCOST) | MSUB.024 |
| | PMIX = PRM | MSUB.025 |
| | M2 = M**2 | MSUB.026 |
| | INFS = 1 | MSUB.027 |
| C | CHECK FOR ILLEGAL INPUT | MSUB.028 |
| | IF (N) 1304, 1304, 1371 | MSUB.029 |
| | 1371 IF (M - MF) 1304, 1304, 1372 | MSUB.030 |
| | 1372 IF (MF - MC) 1304, 1304, 1373 | MSUB.031 |
| | 1373 IF (MC) 1304 , 1304, 1374 | MSUB.032 |
| | 1374 IF (ME - M) 1304, 1375, 1375 | MSUB.033 |
| | 1375 IF(MOD (INFLAG, 4) - 1) 1400, 1320, 100 | MSUB.034 |
| C | | MSUB.035 |
| C | | MSUB.036 |
| C | NEW 1 STARTS PHASE C4E | MSUB.037 |
| C | C****SUBROUTINE NEW (M,N, JH, KB, A, B, MF, ME) | MSUB.038 |
| C | | MSUB.039 |
| C | INITIATE | MSUB.040 |
| | 1400 DO 1401 I = 1, M | MSUB.041 |
| | 1401 JH(I) = 0 | MSUB.042 |
| C | INSTALL SINGLETONS | MSUB.043 |
| | KT = 0 | MSUB.044 |
| | DO 1402 J = 1, N | MSUB.045 |
| | KB(J) = 0 | MSUB.046 |
| | MM = KT + MF | MSUB.047 |
| | LL = KT + M | MSUB.048 |
| C | TALLY ENTRIES IN CONSTRAINTS | MSUB.049 |
| | KQ = 0 | MSUB.050 |
| | DO 1403 L = MM , LL | MSUB.051 |
| | IF (A(L)) 1404, 1403, 1404 | MSUB.052 |
| 1404 | KQ = KQ+1 | MSUB.053 |
| | LQ = L | MSUB.054 |
| 1403 | CONTINUE | MSUB.055 |
| C | CHECK WHETHER J IS CANDIDATE | MSUB.056 |
| | IF (KQ - 1) 1402, 1405, 1402 | MSUB.057 |
| 1405 | IA = LQ- KT | MSUB.058 |
| | IF (JH(IA)) 1402, 1406, 1402 | MSUB.059 |
| 1406 | IF (A(LQ)*B(IA)) 1402, 1407, 1407 | MSUB.060 |
| C | J IS CANDIDATE. INSTALL | MSUB.061 |

AFML-TR-67-121
PART III

| | |
|---|----------|
| 1407 JH(IA) = J | MSUB.062 |
| KB(J) = IA | MSUB.063 |
| 1402 KT = KT + ME | MSUB.064 |
| C | MSUB.065 |
| C**END OF NEW | MSUB.066 |
| C | MSUB.067 |
| C | MSUB.068 |
| 1320 CONTINUE | MSUB.069 |
| C | MSUB.070 |
| C VER 1 FORMS INVERSE FROM KB | MSUB.071 |
| C*****SUBROUTINE VER (A, B, JH, X, E, KB, Y, IOFIX, TPIV, M2) | MSUB.072 |
| C | MSUB.073 |
| C INITIATE | MSUB.074 |
| 1100 ASSIGN 1102 TO KPIV | MSUB.075 |
| ASSIGN 1114 TO KJMY | MSUB.076 |
| IF (LA) 1121, 1121, 1122 | MSUB.077 |
| 1121 INVC = 0 | MSUB.078 |
| 1122 NUMVR = NUMVR + 1 | MSUB.079 |
| DO 1101 I = 1, M2 | MSUB.080 |
| 1101 E(I)=0. | MSUB.081 |
| MM=1 | MSUB.082 |
| DO 1113 I = 1, M | MSUB.083 |
| E(MM) = 1.0 | MSUB.084 |
| X(I) = B(I) | MSUB.085 |
| 1113 MM = MM + M + 1 | MSUB.086 |
| DO 1110 I = MF, M | MSUB.087 |
| IF (JH(I)) 1111, 1110, 1111 | MSUB.088 |
| 1111 JH(I) = 12345 | MSUB.089 |
| 1110 CONTINUE | MSUB.090 |
| INFS = 1 | MSUB.091 |
| C FORM INVERSE | MSUB.092 |
| DO 1102 JT= 1, N | MSUB.093 |
| IF (KB(JT)) 600, 1102, 600 | MSUB.094 |
| C 600 CALL JMY (JT, A, E, M, Y) | MSUB.095 |
| C CHOOSE PIVOT | MSUB.096 |
| 1114 TY = 0. | MSUB.097 |
| DO 1104 I = MF, M | MSUB.098 |
| IF (JH(I) - 12345) 1104, 1105, 1104 | MSUB.099 |
| 1105 IF (ABS (Y(I)) - TY) 1104, 1104, 1106 | MSUB.100 |
| 1106 IR = I | MSUB.101 |
| TY = ABS (Y(I)) | MSUB.102 |
| 1104 CONTINUE | MSUB.103 |
| C TEST PIVOT | MSUB.104 |
| IF (TY - TPIV) 1107, 1108, 1108 | MSUB.105 |
| C BAD PIVOT, ROW IR, COLUMN JT | MSUB.106 |
| 1107 KB(JT)= 0 | MSUB.107 |
| GO TO 1102 | MSUB.108 |
| C PIVOT | MSUB.109 |
| 1108 JH(IR) = JT | MSUB.110 |
| KB(JT) = IR | MSUB.111 |
| GO TO 900 | MSUB.112 |
| C 900 CALL PIV (IR, Y, M, E, Z, X) | MSUB.113 |
| 1102 CONTINUE | MSUB.114 |
| C RESET ARTIFICIALS | MSUB.115 |
| DO 1109 I = 1, M | MSUB.116 |
| IF (JH(I) - 12345) 1109, 1112, 1109 | MSUB.117 |
| 1112 JH(I) = 0 | MSUB.118 |
| 1109 CONTINUE | MSUB.119 |
| C**END OF VER | MSUB.120 |
| C | MSUB.121 |
| C | MSUB.122 |

AFML-TR-67-121
PART III

| | |
|---|----------|
| 100 ASSIGN 705 TO NDEL | MSUB.123 |
| ASSIGN 1000 TO KJMY | MSUB.124 |
| ASSIGN 221 TO KPIV | MSUB.125 |
| C | MSUB.126 |
| C | MSUB.127 |
| C | MSUB.128 |
| C XCK 1 X CHECKER | MSUB.129 |
| C*****SUBROUTINE XCK (M, MF, JH, X, TZERO, JIN) | MSUB.130 |
| C | MSUB.131 |
| C RESET X AND CHECK FOR INFEASIBILITIES | MSUB.132 |
| 1200 JIN = 0 | MSUB.133 |
| NEG = 0 | MSUB.134 |
| DO 1201 I = MF, M | MSUB.135 |
| IF (ABS (X(I)) - TZERO) 1202, 1203, 1203 | MSUB.136 |
| 1202 X(I) = 0.0 | MSUB.137 |
| GO TO 1201 | MSUB.138 |
| 1203 IF (X(I)) 1208, 1201, 1205 | MSUB.139 |
| 1205 IF (JH(I)) 1201, 1206, 1201 | MSUB.140 |
| 1208 NEG = 1 | MSUB.141 |
| 1206 JIN = 1 | MSUB.142 |
| 1201 CONTINUE | MSUB.143 |
| C**END OF XCK | MSUB.144 |
| C | MSUB.145 |
| C | MSUB.146 |
| C CHECK CHANGE OF PHASE.. GO BACK TO INVERT IF GONE INFEAS. | MSUB.147 |
| IF (INFS - JIN) 1320, 500, 200 | MSUB.148 |
| C BECOME FEASIBLE | MSUB.149 |
| 200 INFS = 0 | MSUB.150 |
| 201 PMIX = 0.0 | MSUB.151 |
| C | MSUB.152 |
| C GET 1 GET PRICES | MSUB.153 |
| C*****SUBROUTINE GET (M, MC, MF, JH, X, P, E, INFS, PMIX) | MSUB.154 |
| C | MSUB.155 |
| 500 MM = MC | MSUB.156 |
| C PRIMAL PRICES | MSUB.157 |
| 502 DO 503 J = 1, M | MSUB.158 |
| P(J) = E(MM) | MSUB.159 |
| 503 MM = MM + M | MSUB.160 |
| IF (INFS) 501, 599, 501 | MSUB.161 |
| C COMPOSITE PRICES | MSUB.162 |
| 501 DO 504 J = 1, M | MSUB.163 |
| 504 P(J) = P(J)* PMIX | MSUB.164 |
| DO 505 I = MF, M | MSUB.165 |
| MM = I | MSUB.166 |
| IF (X(I)) 506, 507, 507 | MSUB.167 |
| 506 DO 508 J = 1, M | MSUB.168 |
| P(J) = P(J) + E(MM) | MSUB.169 |
| 508 MM = MM + M | MSUB.170 |
| GO TO 505 | MSUB.171 |
| 507 IF (JH(I)) 505, 509, 505 | MSUB.172 |
| 509 DO 510 J = 1, M | MSUB.173 |
| P(J) = P(J) - E(MM) | MSUB.174 |
| 510 MM = MM + M | MSUB.175 |
| 505 CONTINUE | MSUB.176 |
| C | MSUB.177 |
| 599 CONTINUE | MSUB.178 |
| C**END OF GET | MSUB.179 |
| C | MSUB.180 |
| C | MSUB.181 |
| C | MSUB.182 |
| C MIN MIN D-J. SELECTS COLUMN TO ENTER BASIS | MSUB.183 |

AFML-TR-67-121
PART III

| | |
|--|----------|
| C*****SUBROUTINE MIN (JT, N, M, A, P, KB, ME, ICOST) | MSUB.184 |
| C | MSUB.185 |
| 700 JT = 0 | MSUB.186 |
| BB = ICOST | MSUB.187 |
| C | MSUB.188 |
| 701 DO 702 JM = 1, N | MSUB.189 |
| C SKIP COLUMNS IN BASIS | MSUB.190 |
| 703 IF (KB(JM)) 702, 300, 702 | MSUB.191 |
| C 300 CALL DEL (JM, DT, M, A, P) | MSUB.192 |
| 705 IF (DT - BB) 708, 702, 702 | MSUB.193 |
| 708 BB = DT | MSUB.194 |
| JT = JM | MSUB.195 |
| 702 CONTINUE | MSUB.196 |
| C | MSUB.197 |
| C**END OF MIN | MSUB.198 |
| C | MSUB.199 |
| IF (JT) 203, 203, 600 | MSUB.200 |
| C ALL COSTS NON-NEGATIVE... K = 3 OR 4 | MSUB.201 |
| 203 K = 3 + INFS | MSUB.202 |
| GO TO 257 | MSUB.203 |
| C | MSUB.204 |
| C NORMAL CYCLE | MSUB.205 |
| C | MSUB.206 |
| C JMY 1 J MULTIPLY. BASIS INVERSE * COLUMN JT | MSUB.207 |
| C*****SUBROUTINE JMY (JT, A, E, M, Y, ME) | MSUB.208 |
| C | MSUB.209 |
| 600 DO 610 I = 1, M | MSUB.210 |
| 610 Y(I) = 0. | MSUB.211 |
| LP = JT*ME - ME | MSUB.212 |
| LL = 0 | MSUB.213 |
| DO 605 I = 1, M | MSUB.214 |
| LP = LP + 1 | MSUB.215 |
| IF (A(LP)) 601, 602, 601 | MSUB.216 |
| 601 DO 606 J = 1, M | MSUB.217 |
| LL = LL + 1 | MSUB.218 |
| 606 Y(J) = Y(J) + A(LP) * E(LL) | MSUB.219 |
| GO TO 605 | MSUB.220 |
| 602 LL = LL + M | MSUB.221 |
| 605 CONTINUE | MSUB.222 |
| C | MSUB.223 |
| 699 GO TO KJMY , (1000 , 1114 , 1392) | MSUB.224 |
| C**END OF JMY | MSUB.225 |
| C | MSUB.226 |
| C | MSUB.227 |
| C ROW 1 ROW SELECTION--COMPOSITE | MSUB.228 |
| C*****SUBROUTINE ROW (IR, M, MF, JH, X, Y, IPIV) | MSUB.229 |
| C | MSUB.230 |
| C AMONG EQS. WITH X=0, FIND MAX ABS(Y) AMONG ARTIFICIALS, OR, IF NONE, | MSUB.231 |
| C GET MAX POSITIVE Y(I) AMONG REALS. | MSUB.232 |
| 1000 IR = 0 | MSUB.233 |
| AA = 0.0 | MSUB.234 |
| IA = 0 | MSUB.235 |
| DO 1050 I = MF, M | MSUB.236 |
| IF (X(I)) 1050, 1041, 1050 | MSUB.237 |
| 1041 YI = ABS (Y(I)) | MSUB.238 |
| IF (YI - IPIV) 1050, 1050, 1042 | MSUB.239 |
| 1042 IF (JH(I)) 1043, 1044, 1043 | MSUB.240 |
| 1043 IF (IA) 1050, 1048, 1050 | MSUB.241 |
| 1048 IF (Y(I)) 1050, 1050, 1045 | MSUB.242 |
| 1044 IF (IA) 1045, 1046, 1045 | MSUB.243 |
| 1045 IF (YI - AA) 1050, 1050, 1047 | MSUB.244 |
| 1046 IA = 1 | |

AFML-TR-67-121
PART III

| | |
|---|----------|
| 1047 AA = YI | MSUB.245 |
| IR = I | MSUB.246 |
| 1050 CONTINUE | MSUB.247 |
| IF (IR)1099,1001,1099 | MSUB.248 |
| 1001 AA = 1.0E+20 | MSUB.249 |
| C FIND MIN. PIVOT AMONG POSITIVE EQUATIONS | MSUB.250 |
| DO 1010 IT = MF , M | MSUB.251 |
| IF (Y(IT) - TPIV) 1010, 1010, 1002 | MSUB.252 |
| 1002 IF (X(IT)) 1010, 1010, 1003 | MSUB.253 |
| 1003 XY = X(IT) / Y(IT) | MSUB.254 |
| IF (XY - AA) 1004, 1005, 1010 | MSUB.255 |
| 1005 IF (JH(IT)) 1010, 1004, 1010 | MSUB.256 |
| 1004 AA = XY | MSUB.257 |
| IR = IT | MSUB.258 |
| 1010 CONTINUE | MSUB.259 |
| IF (NEG) 1016, 1099, 1016 | MSUB.260 |
| C FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS THAN THE | MSUB.261 |
| C MINIMUM X/Y IN THE POSITIVE EQUATIONS, THAT HAS THE LARGEST ABSF(Y) | MSUB.262 |
| 1016 BB = - TPIV | MSUB.263 |
| DO 1030 I = MF , M | MSUB.264 |
| IF (X(I)) 1012, 1030, 1030 | MSUB.265 |
| 1012 IF (Y(I) - BB) 1022, 1030, 1030 | MSUB.266 |
| 1022 IF (Y(I) * AA - X(I)) 1024, 1024, 1030 | MSUB.267 |
| 1024 BB = Y(I) | MSUB.268 |
| IR = I | MSUB.269 |
| 1030 CONTINUE | MSUB.270 |
| 1099 CONTINUE | MSUB.271 |
| C**END OF ROW | MSUB.272 |
| C | MSUB.273 |
| C TEST PIVOT | MSUB.274 |
| 206 IF(IR) 207, 207, 210 | MSUB.275 |
| C NO PIVOT | MSUB.276 |
| 207 K = 5 | MSUB.277 |
| 257 IF (PMIX) 201, 400, 201 | MSUB.278 |
| C ITERATION LIMIT FOR CUT OFF | MSUB.279 |
| 210 IF (ITER -NCUT) 900, 160, 160 | MSUB.280 |
| C PIVOT FOUND | MSUB.281 |
| C | MSUB.282 |
| C PIV 1 PIVOT. PIVOTS ON GIVEN ROW | MSUB.283 |
| C*****SUBROUTINE PIV (IR, Y, M, E, X, NUMPV, TECOL) | MSUB.284 |
| C LEAVE TRANSFORMED COLUMN IN Y(I) | MSUB.285 |
| C | MSUB.286 |
| 900 NUMPV = NUMPV + 1 | MSUB.287 |
| YI = -Y(IR) | MSUB.288 |
| Y(IR) = -1. | MSUB.289 |
| LL = 0 | MSUB.290 |
| C TRANSFORM INVERSE | MSUB.291 |
| 903 DO 904 L = IR, M2, M | MSUB.292 |
| IF (F(L)) 905, 914, 905 | MSUB.293 |
| 914 LL = LL + M | MSUB.294 |
| GO TO 904 | MSUB.295 |
| 905 XY = E(L) / YI | MSUB.296 |
| E(L) = 0. | MSUB.297 |
| DO 906 I = 1, M | MSUB.298 |
| LL = LL + 1 | MSUB.299 |
| 906 E(LL) = E(LL) + XY* Y(I) | MSUB.300 |
| 904 CONTINUE | MSUB.301 |
| C TRANSFORM X | MSUB.302 |
| XY = X(IR) / YI | MSUB.303 |
| X(IR) = 0. | MSUB.304 |
| DO 908 I = 1, M | MSUB.305 |

AFML-TR-67-121
PART III

| | |
|---|----------|
| 908 X(I) = X(I) +XY* Y(I) | MSUB.306 |
| C RESTORE Y(IR) | MSUB.307 |
| Y(IR) = -YI | MSUB.308 |
| C | MSUB.309 |
| 999 GO TO KPIV , (221, 1102) | MSUB.310 |
| C**END OF PIV | MSUB.311 |
| C | MSUB.312 |
| 221 IA = JH(IR) | MSUB.313 |
| IF (IA) 213, 213, 214 | MSUB.314 |
| 214 KB(IA) = 0 | MSUB.315 |
| 213 KB(JT) = IR | MSUB.316 |
| JH(IR) = JT | MSUB.317 |
| LA = 0 | MSUB.318 |
| ITER = ITER +1 | MSUB.319 |
| INVC = INVC +1 | MSUB.320 |
| C INVERSION FREQUENCY | MSUB.321 |
| IF (INVC - NVER) 1200, 1320, 1200 | MSUB.322 |
| C CUT OFF ... TOO MANY ITERATIONS | MSUB.323 |
| 160 K = 6 | MSUB.324 |
| C | MSUB.325 |
| C | MSUB.326 |
| C ERR 1 ERROR CHECK. COMPARES AX WITH B, PA WITH ZERO | MSUB.327 |
| C*****SUBROUTINE ERR (M, A, B, TERR, JH, X, P, Y, ME, LA) | MSUB.328 |
| C | MSUB.329 |
| C STORE AX-B AT Y | MSUB.330 |
| 400 ASSIGN 410 TO NDEL | MSUB.331 |
| DO 401 I = 1, M | MSUB.332 |
| 401 Y(I) =-B(I) | MSUB.333 |
| DO 402 I = 1, M | MSUB.334 |
| JA = JH(I) | MSUB.335 |
| IF (JA) 403, 402, 403 | MSUB.336 |
| 403 IA =ME* (JA-1) | MSUB.337 |
| DO 405 IT = 1, M | MSUB.338 |
| IA = IA + 1 | MSUB.339 |
| IF(A(IA)) 415, 405, 415 | MSUB.340 |
| 415 Y(IT) =Y(IT) +X(I) * A(IA) | MSUB.341 |
| 405 CONTINUE | MSUB.342 |
| 402 CONTINUE | MSUB.343 |
| C FIND SUM AND MAXIMUM OF ERRORS | MSUB.344 |
| DO 481 I = 1, M | MSUB.345 |
| YI = Y(I) | MSUB.346 |
| IF (JH(I)) 472, 471, 472 | MSUB.347 |
| 471 YI = YI + X(I) | MSUB.348 |
| 472 TERR(LA+1) = TERR(LA+1) + ABS (YI) | MSUB.349 |
| IF (ABS (TERR(LA+2))- ABS (YI)) 482, 481, 481 | MSUB.350 |
| 482 TERR(LA+2) = YI | MSUB.351 |
| 481 CONTINUE | MSUB.352 |
| C STORE P TIMES BASIS AT DT | MSUB.353 |
| DO 411 I = 1, M | MSUB.354 |
| JM = JH(I) | MSUB.355 |
| IF (JM) 300 , 411 , 300 | MSUB.356 |
| C 300 CALL DEL (JM, DT, M, A, P) | MSUB.357 |
| 410 TERR(LA+3) = TERR(LA +3) + ABS (DT) | MSUB.358 |
| IF (ABS (TERR(LA+4)) - ABS (DT)) 413, 411, 411 | MSUB.359 |
| 413 TERR(LA+4) = DT | MSUB.360 |
| 411 CONTINUE | MSUB.361 |
| C**END OF ERR | MSUB.362 |
| C | MSUB.363 |
| C | MSUB.364 |
| IF (LA) 193, 191, 193 | MSUB.365 |
| 191 LA = 4 | MSUB.366 |

AFML-TR-67-121
PART III

| | |
|---|----------|
| IF (INFLAG - 4) 1320, 193, 193 | MSUB.367 |
| 193 IF (K-5) 1392, 194, 1392 | MSUB.368 |
| 194 ASSIGN 1392 TO KJMY | MSUB.369 |
| GO TO 600 | MSUB.370 |
| C 600 CALL JMY (.) | MSUB.371 |
| C GO TO 1392 | MSUB.372 |
| 1304 K = 7 | MSUB.373 |
| C SET EXIT VALUES | MSUB.374 |
| 1392 DO 1309 I= 1, 8 | MSUB.375 |
| 1309 ERR(I) = TERR(I) | MSUB.376 |
| DO 1329 I = 1, 7 | MSUB.377 |
| 1329 KOUT(I) = IOFIX(I+8) | MSUB.378 |
| RETURN | MSUB.379 |
| C | MSUB.380 |
| C DEL DELTA-JAY. PRICES OUT ONE MATRIX COLUMN | MSUB.381 |
| C*****SUBROUTINE DEL (JM, DT, M, A, P, ME) | MSUB.382 |
| C | MSUB.383 |
| C | MSUB.384 |
| 300 DT = 0. | MSUB.385 |
| LL = (JM - 1) * ME | MSUB.386 |
| 301 DO 303 MM = 1, M | MSUB.387 |
| LL = LL + 1 | MSUB.388 |
| IF (A(LL))304, 303, 304 | MSUB.389 |
| 304 DT = DT + P(MM) * A (LL) | MSUB.390 |
| 303 CONTINUE | MSUB.391 |
| C | MSUB.392 |
| 399 GO TO NDEL , (410 , 705) | MSUB.393 |
| C**END OF DEL | MSUB.394 |
| C | MSUB.395 |
| END | MSUB.396 |

REFERENCES

1. H. Fujita, "Mathematical Theory of Sedimentation Analysis, Academic Press, New York, 1962.
2. T. H. Donnelly, J. Phys. Chem., 70, 1862 (1966).
3. E. T. Adams, Jr., "Molecular Weights and Molecular-Weight Distributions from Sedimentation-Equilibrium Experiments," p. 84, "Characterization of Macromolecular Structure," Publication 1573 National Academy of Sciences, Washington, D. C., 1968.
4. D. A. Lee, (private communication), to be published.
5. M. Gehatia, AFML-TR-67-121, Pt. I.
6. Th. G. Scholte, J. of Polymer Sci., A-2, 6, 91-109 (1968).
7. Th. G. Scholte, J. of Polymer Sci., A-2, 6, 111-127 (1968).
8. M. Gehatia and D. R. Wiff, AFML-TR-67-121, Pt. II.
9. S. I. Gass, "Linear Programming Methods and Applications," 2nd Ed., McGraw-Hill Company, New York, 1964.
10. A. Charnes, W. W. Cooper, and A. Henderson, "Introduction to Linear Programming," John Wiley & Sons, Inc., New York, 1953.
11. G. B. Dantzig, "Maximization of a Linear Function of Variables Subject to Linear Inequalities," Chap. XXI, T. C. Koopman (ed.), "Activity Analysis of Production and Allocation," Cowles Commission Monograph 13, John Wiley & Sons, Inc., New York, 1951.
12. A. N. Tikhonov and V. B. Glasko, USSR Computational Mathematics and Mathematical Physics, 4, 1 (1964).
13. A. N. Tikhonov, Dokl. Akad. Nauk SSSR, 151, 3, 501-504 (1963).
14. A. N. Tikhonov, Dokl. Akad. Nauk. SSSR, 153, 1, 49-52 (1963).

PART III

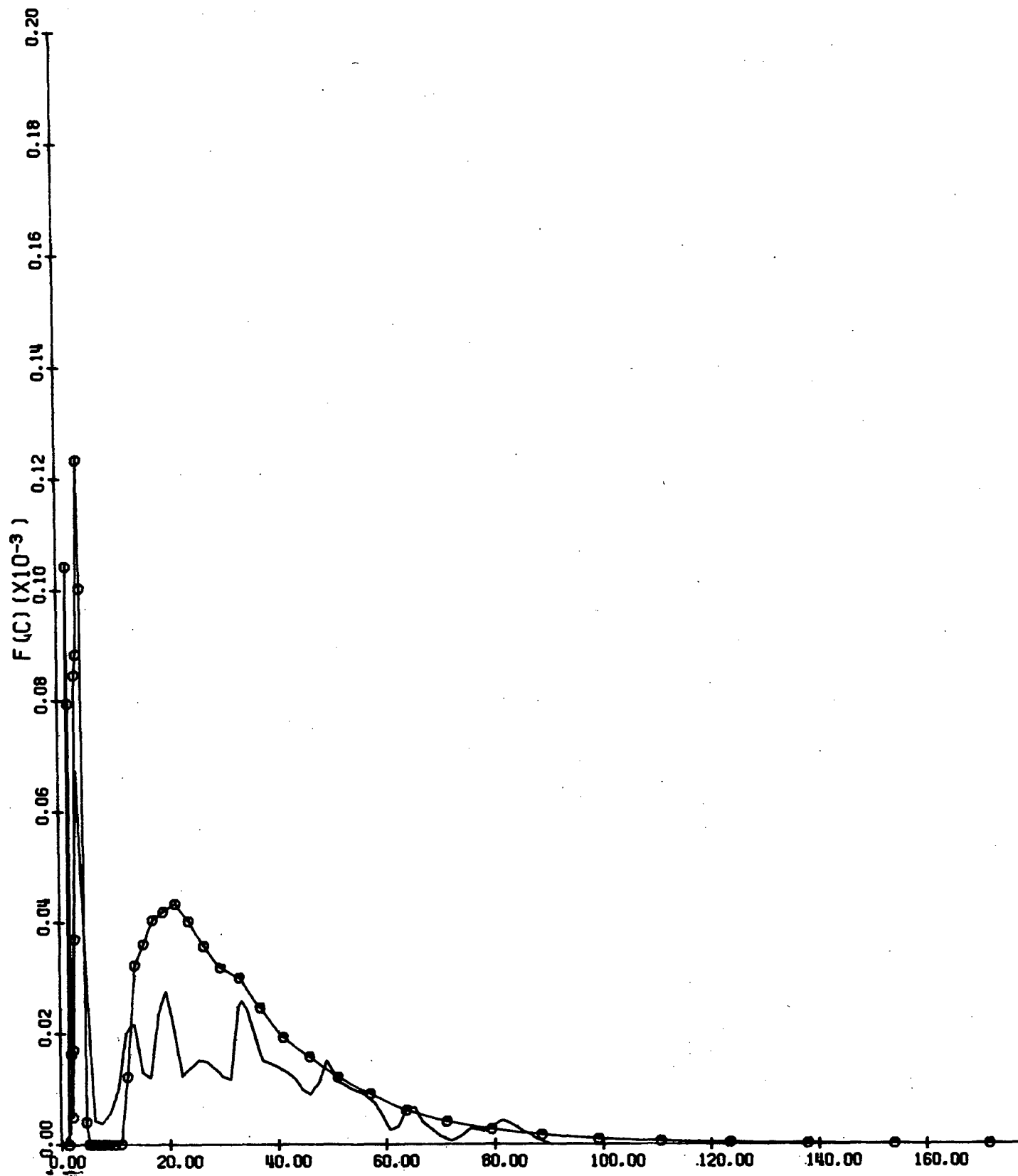
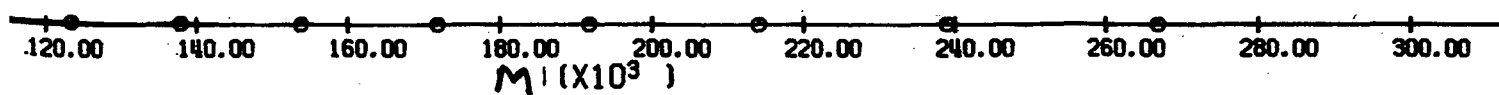


Figure 1. Resulting MWD Using Ten Sets of Molecular Weights a



Ten Sets of Molecular Weights and $g = 3.0$, (Compare with Figures 2-4)

AFML-TR-67-121
PART III

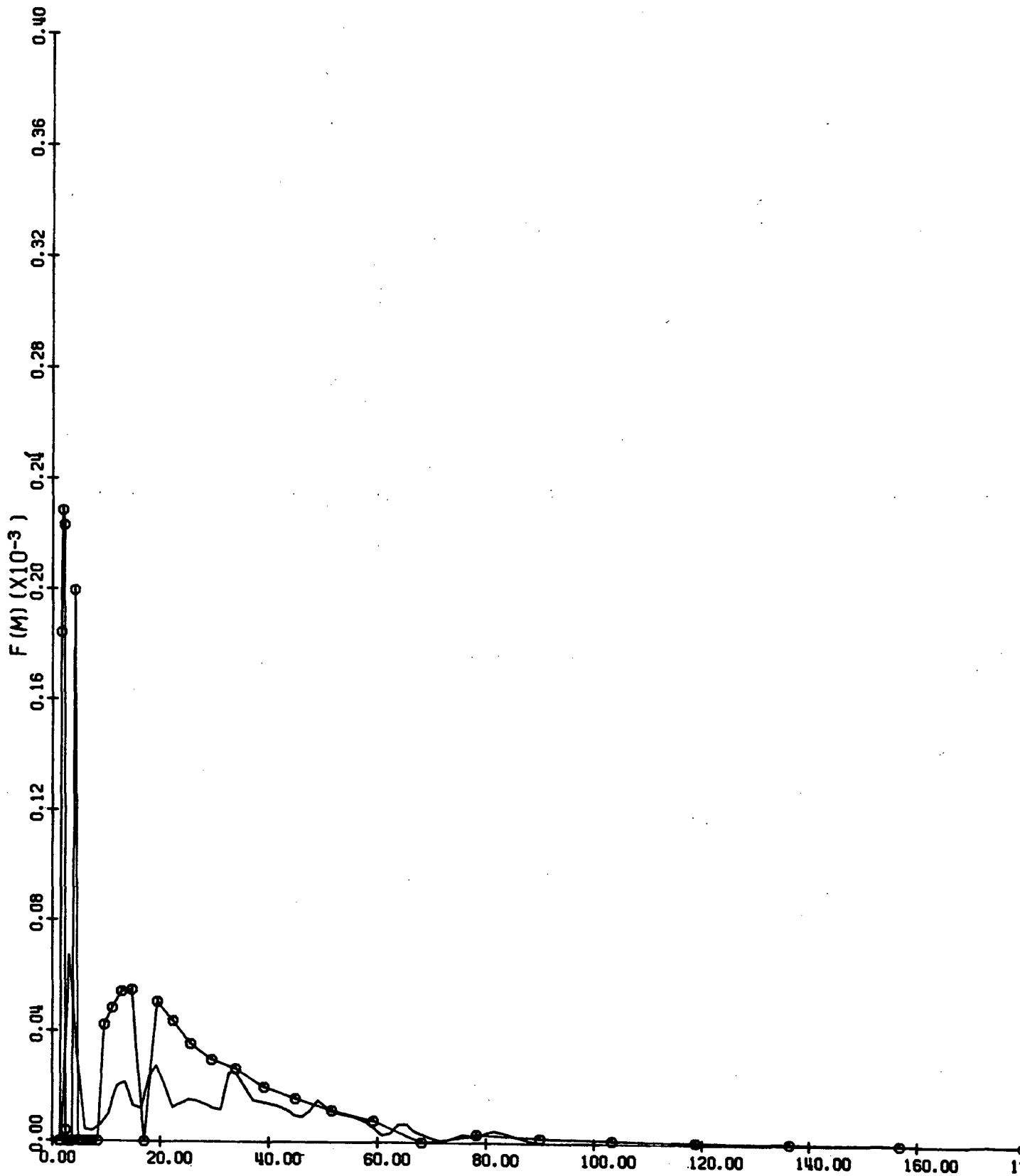
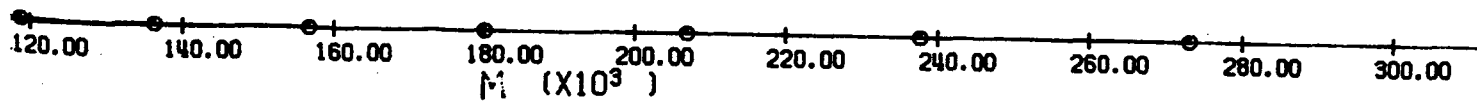


Figure 2. Resulting MWD Using Ten Sets of Molecular Weights and $g =$



Sets of Molecular Weights and $g = 4.0$, (Compare with Figures 1, 3, and 4)

AFML-TR-67-121
PART III

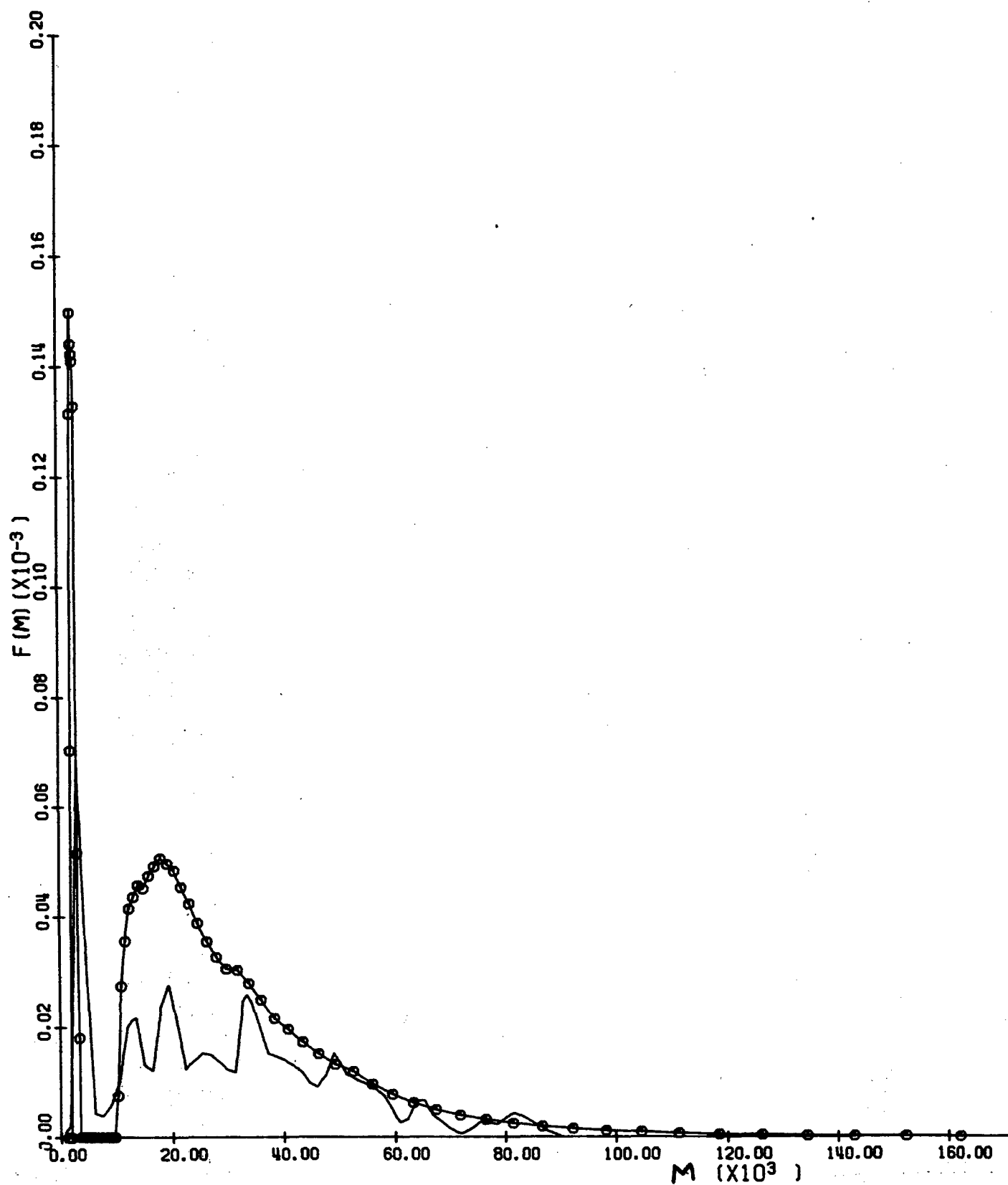
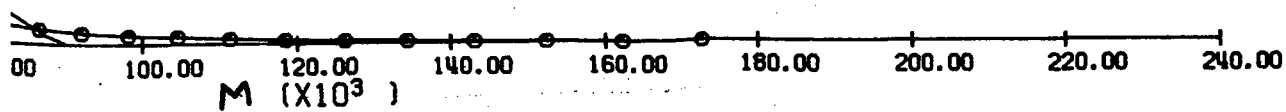


Figure 3. Resulting MWD Using Twenty Sets of Molecular Weights and $g = 3.5$, (Co



enty Sets of Molecular Weights and $g = 3.5$, (Compare with Figures 1, 2, and 4)

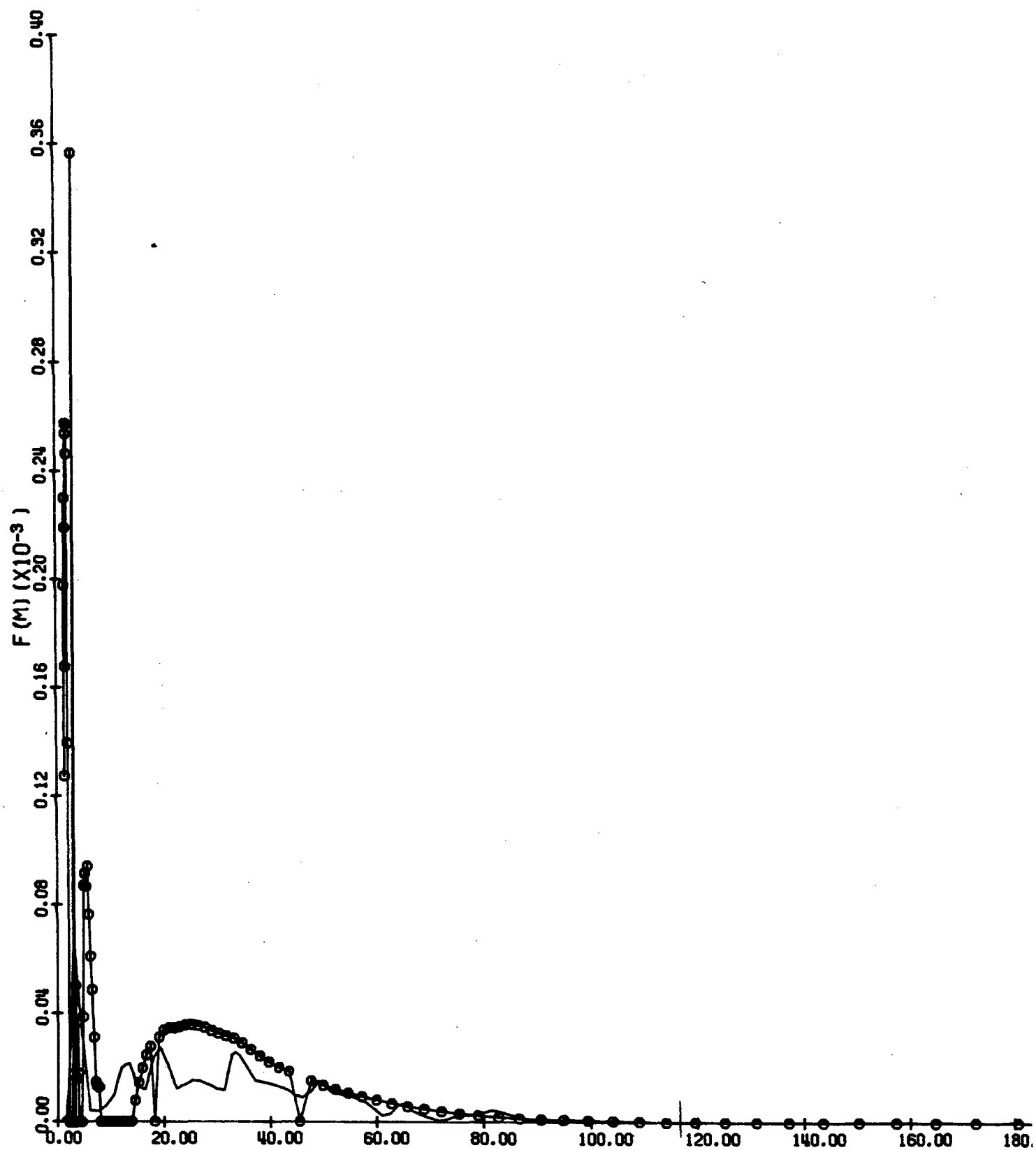
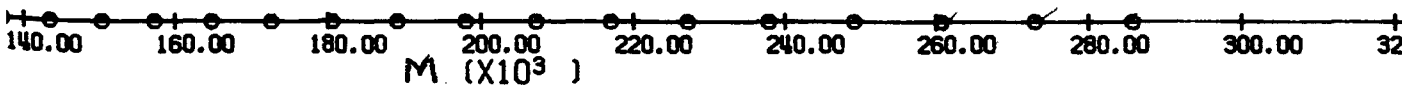


Figure 4. Resulting MWD Using Twenty Sets of Molecular V



enty Sets of Molecular Weights and $g = 2.5$, (Compare with Figures 1-3)

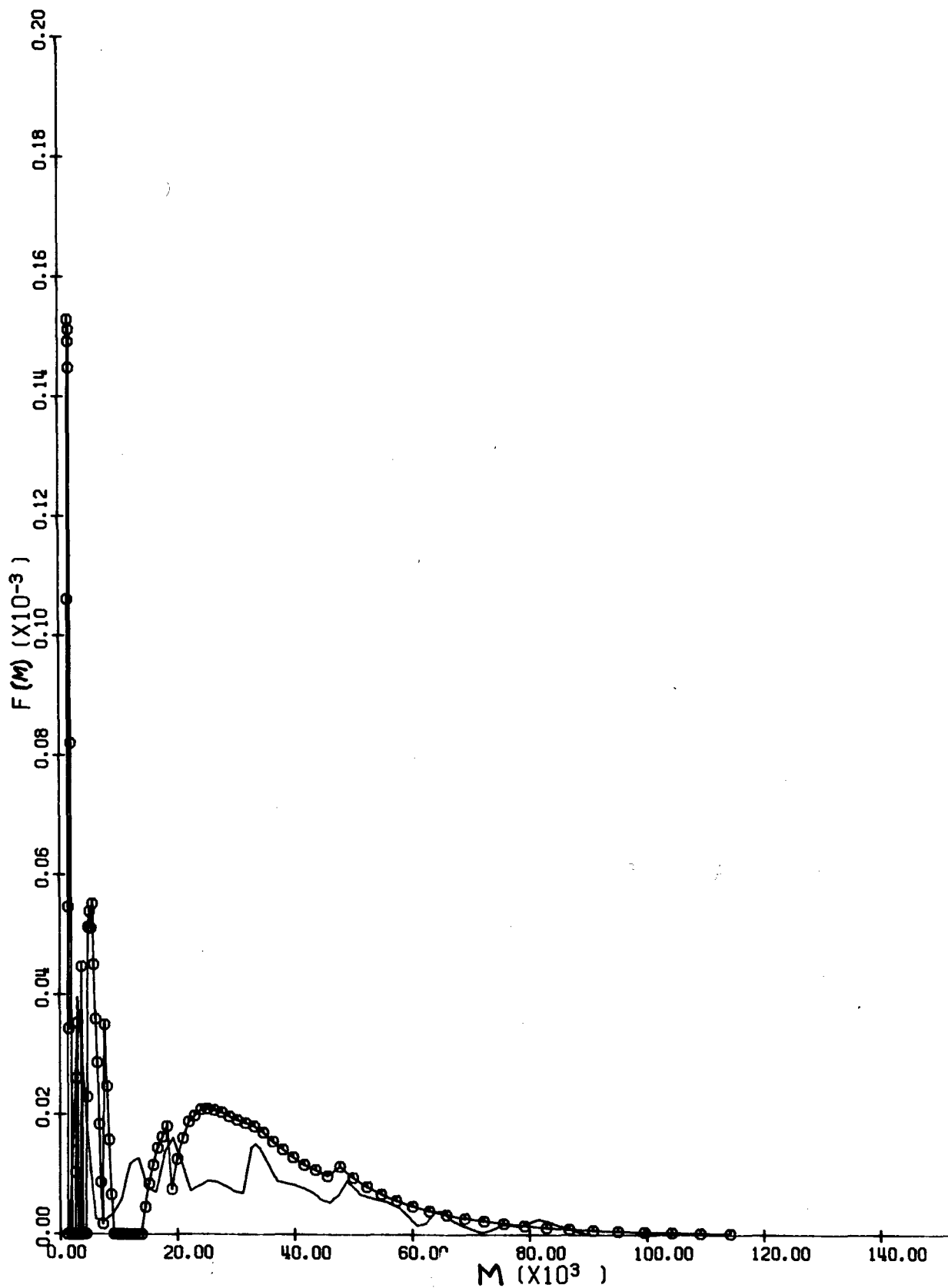


Figure 5. The Effect of Decreasing the MWR on the MWD

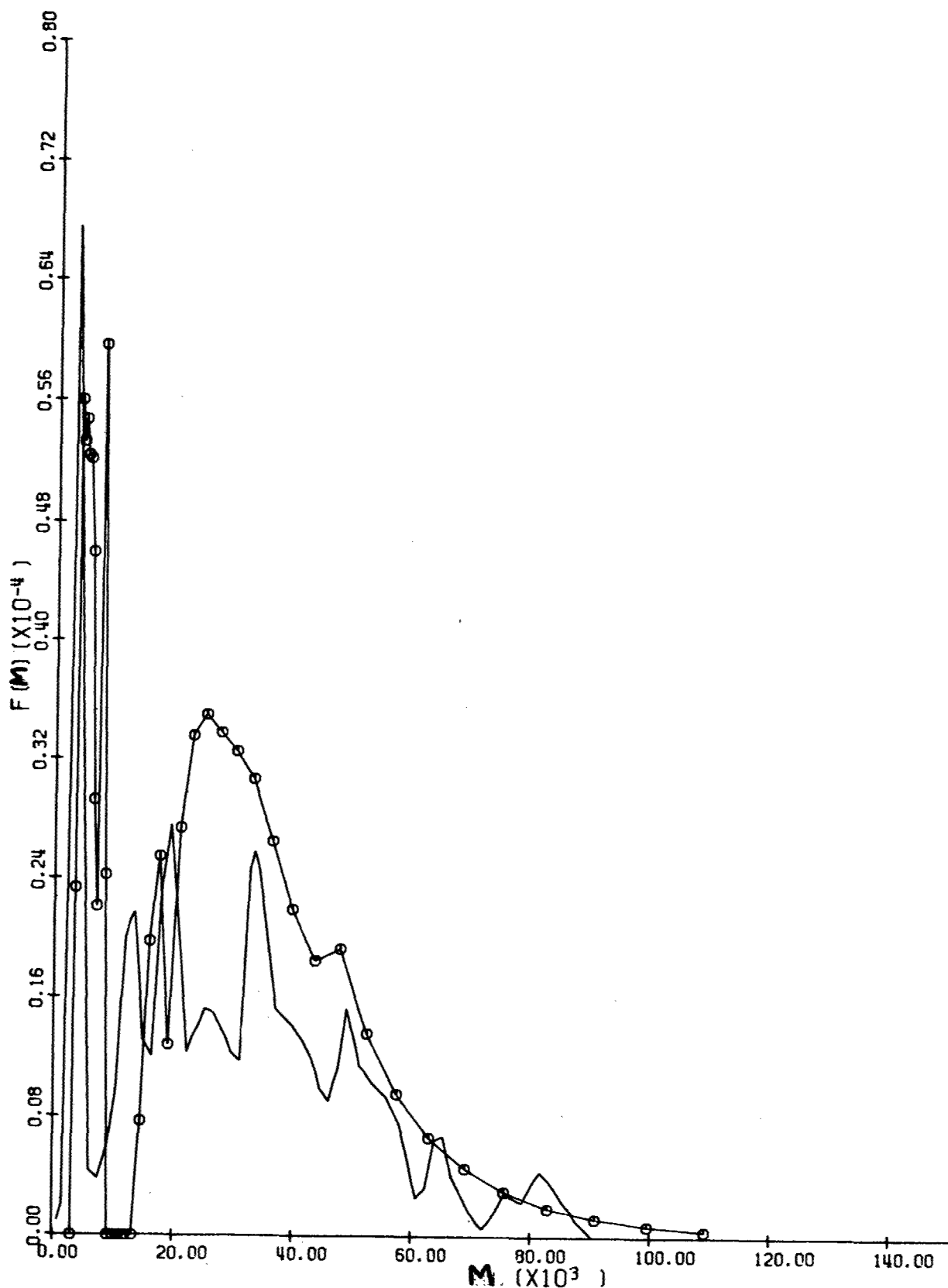


Figure 6. The Effect of Decreasing the MWR and the Number of Molecular Weight Sets on the MWD

AFML-TR-67-121
PART III

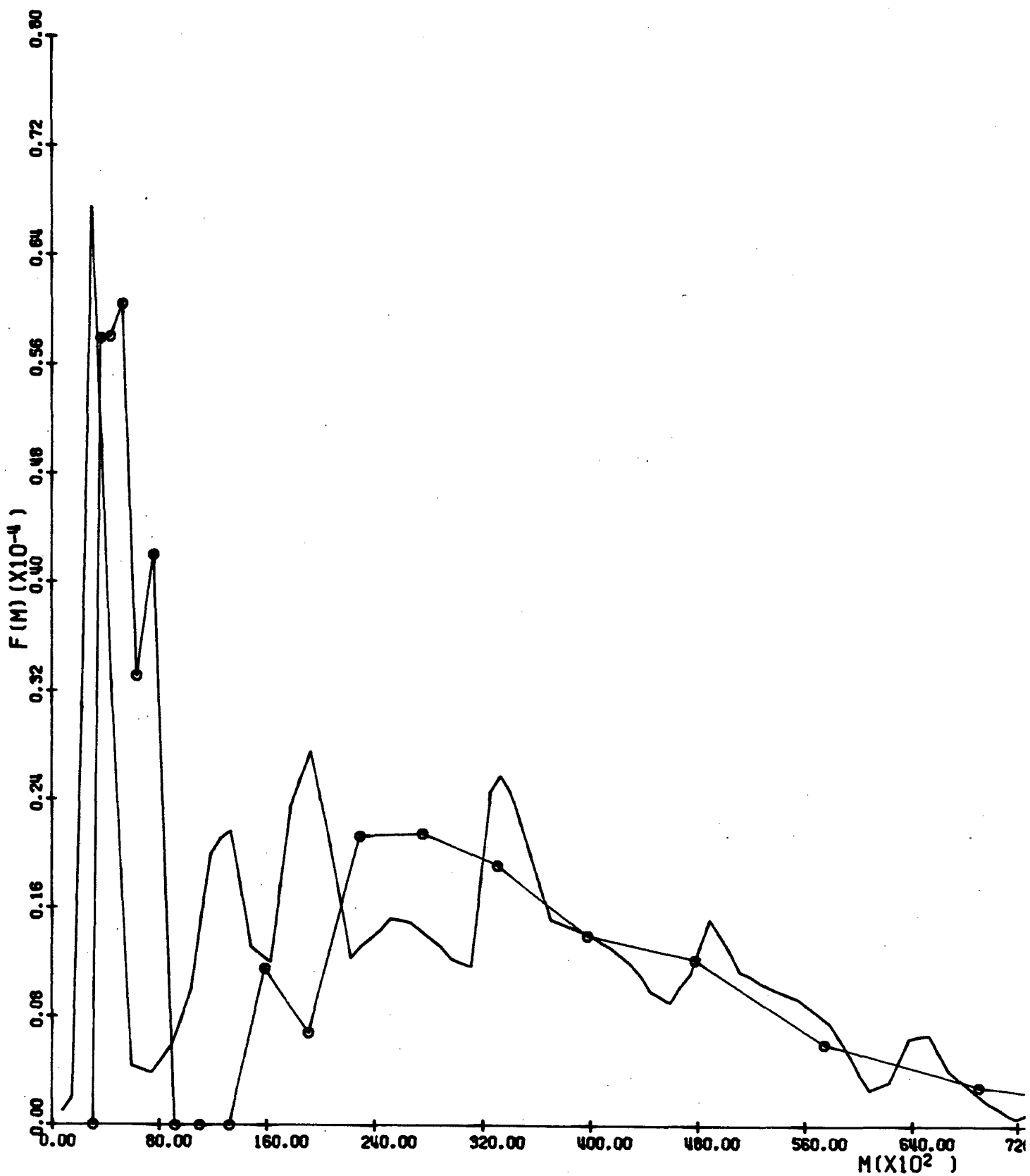
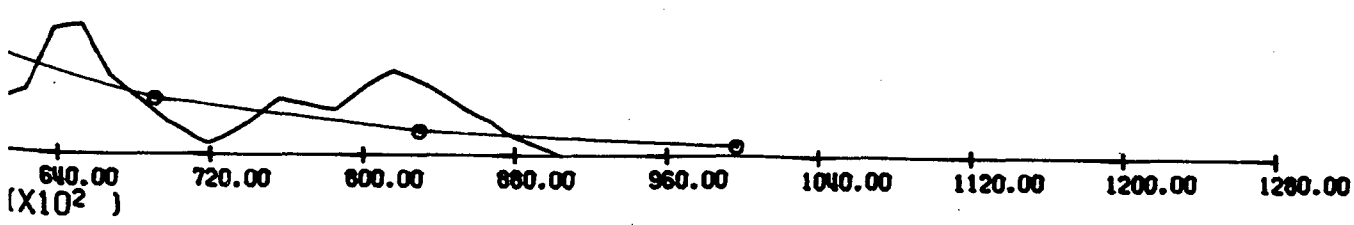


Figure 7. The Effect of Further Decreasing t



or Decreasing the Number of Molecular Weight Sets

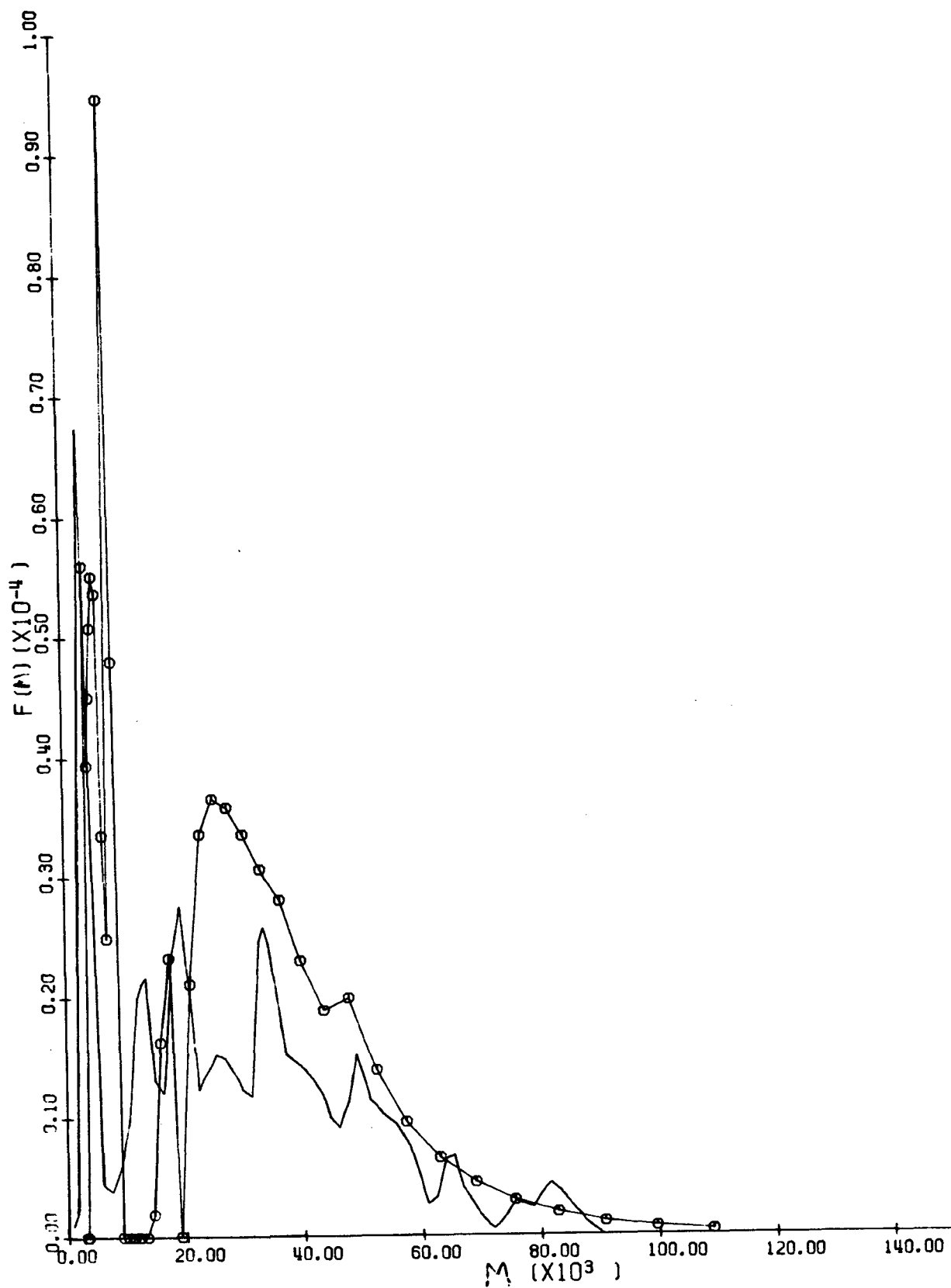


Figure 8. The Effect of Using More Experimental Data on the MWD

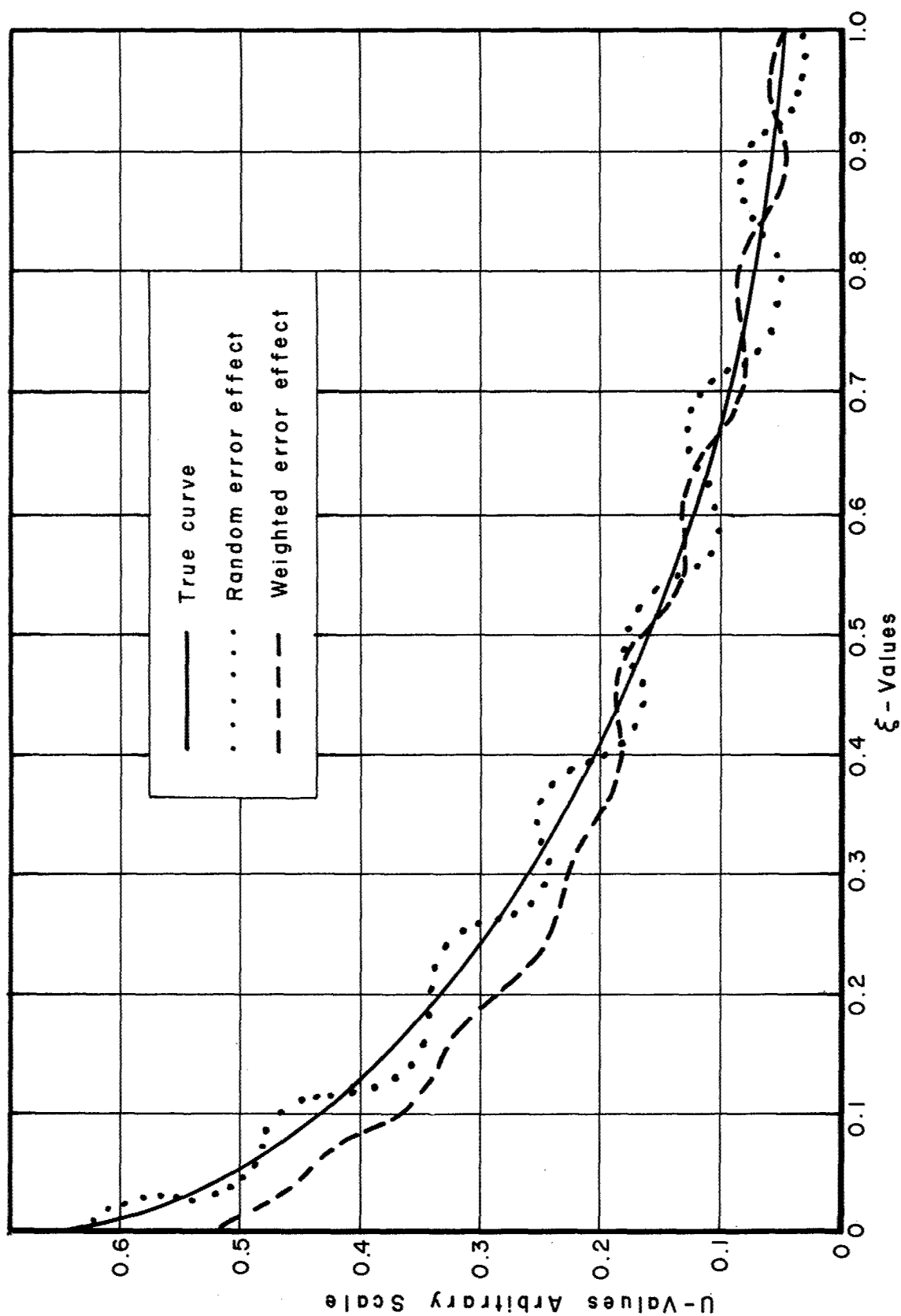


Figure 9. A Typical Curve of Concentration Gradient from One Angular Velocity

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Air Force Materials Laboratory
Wright-Patterson AFB, Ohio 45433

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

EVALUATION OF MOLECULAR WEIGHT FROM EQUILIBRIUM SEDIMENTATION. PART III.
MOLECULAR WEIGHT DISTRIBUTIONS FROM EQUILIBRIUM SEDIMENTATION-DIFFUSION
DATA VIA LINEAR PROGRAMMING

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Summary Report (September 1968 to August 1969)

5. AUTHOR(S) (First name, middle initial, last name)

Robert R. Jurick
Donald R. Wiff
Matatiah Gehatia

6. REPORT DATE

May 1970

7a. TOTAL NO. OF PAGES

68

7b. NO. OF REFS

14

8a. CONTRACT OR GRANT NO.

b. PROJECT NO. 7342

c. Task No. 734203

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

AFML-TR-67-121, Part III

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Air Force Materials Laboratory
Wright-Patterson AFB, Ohio 45433

13. ABSTRACT

Within the past decade easy access to high speed digital computers has renewed interest in deriving a molecular weight distribution from equilibrium sedimentation-diffusion data. One of the computational schemes which appears most promising is the Simplex Method of linear programming. The purpose of this work was to investigate the advantages and limitations of this approach.

It was found that even though inferring a molecular weight distribution from equilibrium sedimentation-diffusion data is mathematically an ill-posed problem, the method of linear programming yields qualitatively a good molecular weight distribution. Also, the method is applicable to the case of one angular velocity.

DD FORM 1473
1 NOV 65

UNCLASSIFIED

Security Classification

Security Classification

14

KEY WORDS

LINK A

LINK B

LINK C

| ROLE |
|-------------------------------|
| 1. Chairman |
| 2. Secretary |
| 3. Member |
| 4. Guest |
| 5. Observer |
| 6. Participant |
| 7. Facilitator |
| 8. Recorder |
| 9. Timekeeper |
| 10. Reporter |
| 11. Debate Moderator |
| 12. Panelist |
| 13. Questioner |
| 14. Answerer |
| 15. Debate Participant |
| 16. Debate Observer |
| 17. Debate Participant |
| 18. Debate Observer |
| 19. Debate Participant |
| 20. Debate Observer |
| 21. Debate Participant |
| 22. Debate Observer |
| 23. Debate Participant |
| 24. Debate Observer |
| 25. Debate Participant |
| 26. Debate Observer |
| 27. Debate Participant |
| 28. Debate Observer |
| 29. Debate Participant |
| 30. Debate Observer |
| 31. Debate Participant |
| 32. Debate Observer |
| 33. Debate Participant |
| 34. Debate Observer |
| 35. Debate Participant |
| 36. Debate Observer |
| 37. Debate Participant |
| 38. Debate Observer |
| 39. Debate Participant |
| 40. Debate Observer |
| 41. Debate Participant |
| 42. Debate Observer |
| 43. Debate Participant |
| 44. Debate Observer |
| 45. Debate Participant |
| 46. Debate Observer |
| 47. Debate Participant |
| 48. Debate Observer |
| 49. Debate Participant |
| 50. Debate Observer |
| 51. Debate Participant |
| 52. Debate Observer |
| 53. Debate Participant |
| 54. Debate Observer |
| 55. Debate Participant |
| 56. Debate Observer |
| 57. Debate Participant |
| 58. Debate Observer |
| 59. Debate Participant |
| 60. Debate Observer |
| 61. Debate Participant |
| 62. Debate Observer |
| 63. Debate Participant |
| 64. Debate Observer |
| 65. Debate Participant |
| 66. Debate Observer |
| 67. Debate Participant |
| 68. Debate Observer |
| 69. Debate Participant |
| 70. Debate Observer |
| 71. Debate Participant |
| 72. Debate Observer |
| 73. Debate Participant |
| 74. Debate Observer |
| 75. Debate Participant |
| 76. Debate Observer |
| 77. Debate Participant |
| 78. Debate Observer |
| 79. Debate Participant |
| 80. Debate Observer |
| 81. Debate Participant |
| 82. Debate Observer |
| 83. Debate Participant |
| 84. Debate Observer |
| 85. Debate Participant |
| 86. Debate Observer |
| 87. Debate Participant |
| 88. Debate Observer |
| 89. Debate Participant |
| 90. Debate Observer |
| 91. Debate Participant |
| 92. Debate Observer |
| 93. Debate Participant |
| 94. Debate Observer |
| 95. Debate Participant |
| 96. Debate Observer |
| 97. Debate Participant |
| 98. Debate Observer |
| 99. Debate Participant |
| 100. Debate Observer |

WT

ROLE

WT

[illegible]

WT

Simplex Method

Security Classification